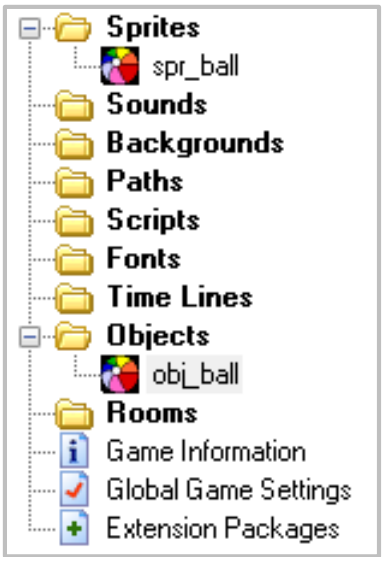
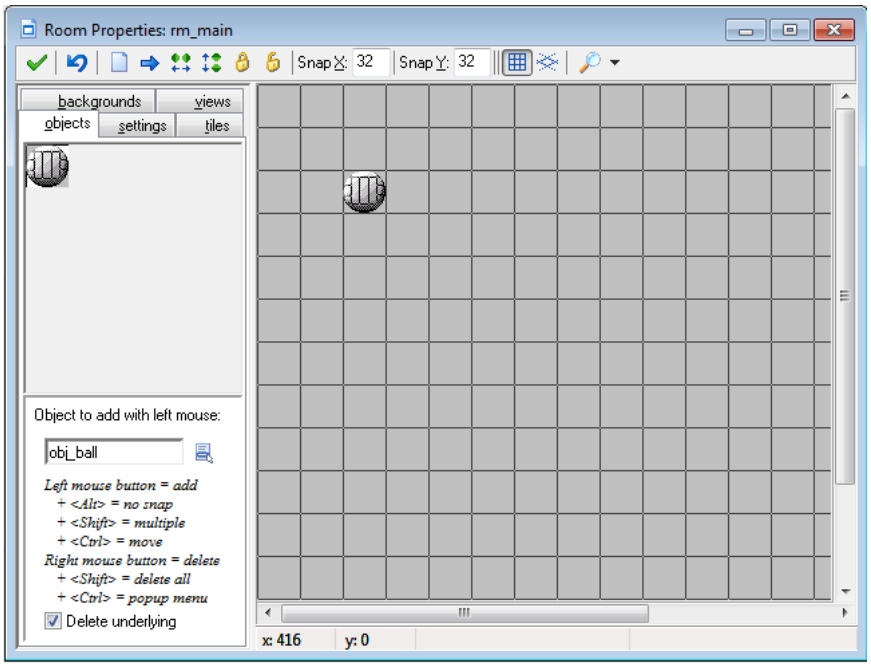
INTRODUCTION TO GAME MAKER:
MOVING OBJECTS AND SETTING PATHS

Before we begin learning how to move objects, we will need to create a new project and add a ball sprite (that we will name **spr_ball**) and a ball object (that we will call **obj_ball**). You can find an image called **ball.png** in the shared directory.



Once you have created both the sprite and the object, create a room and add the ball object anywhere in the room.



MOVE ACTIONS

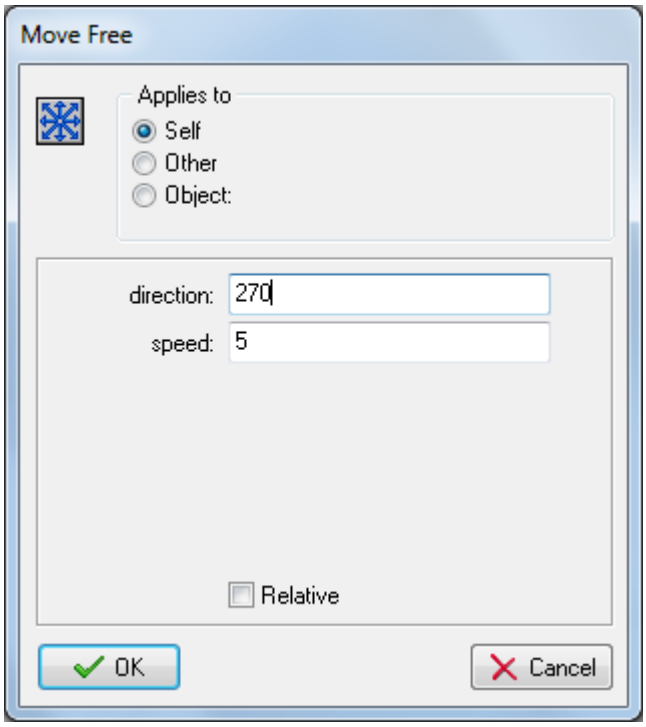
1. MOVE FIXED

- This action starts an object moving in a specified or random direction.
- It includes the speed you want the object to move, which is based on pixels per step.
- You can also specify multiple directions, in which case the object will randomly select which direction to move.
- The middle button stops the object from moving.
- If you select **Relative** for this action, any new motion is added to the current motion. Say, for example, the object is moving up, and then something happens during the game to cause the object to move left. If you select **Relative**, the object will start moving up and to the left.

2. MOVE FREE

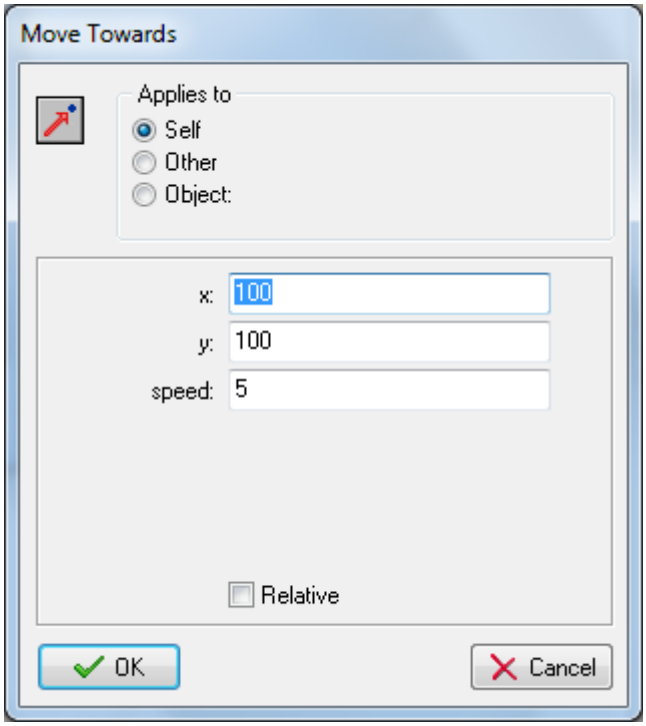
- Use this action to have the object move at a specific angle between 0 and 360 degrees.
- If you use 0, the object moves to the right, a 90-degree angle moves the object up, and so on.

- You can also use **random(360)** to specify a random movement.



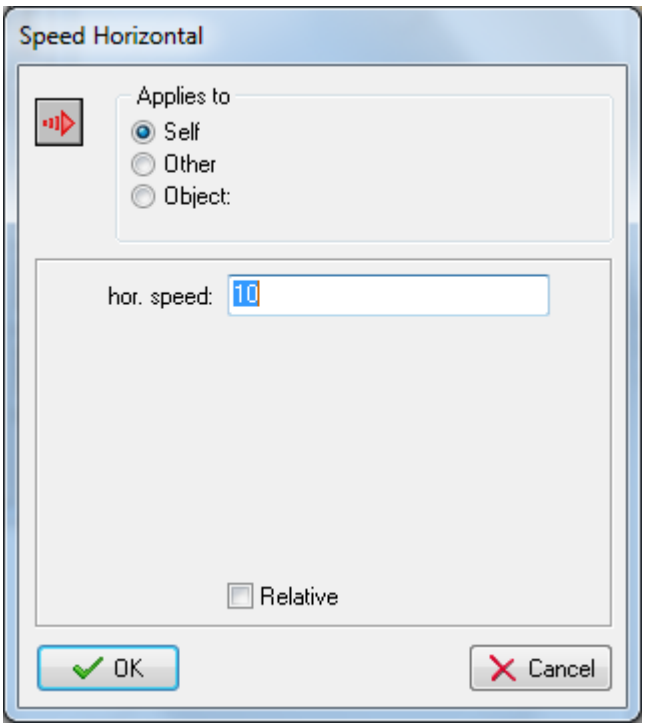
3. MOVE TOWARDS

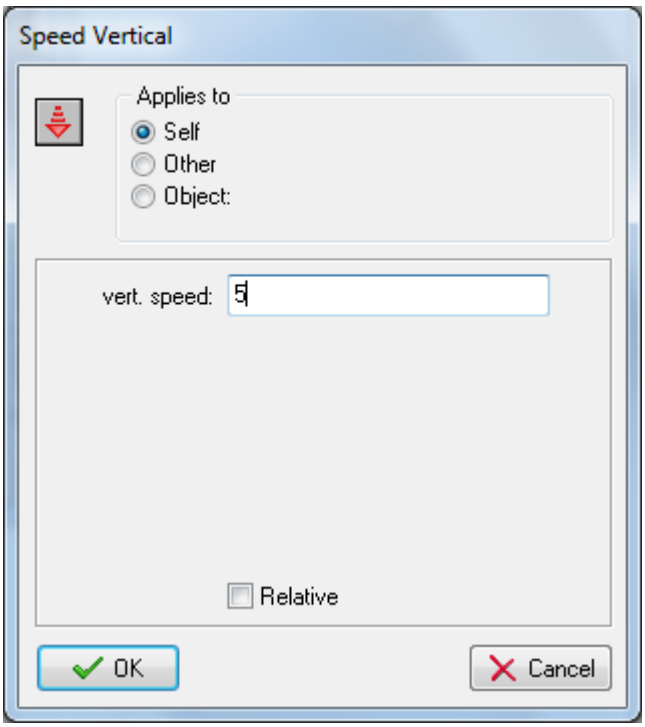
- This action enables you to send an object toward another object.
- You can set the speed of the object and then use the coordinates of another object as the direction for it to move in.



4. SPEED HORIZONTAL and SPEED VERTICAL

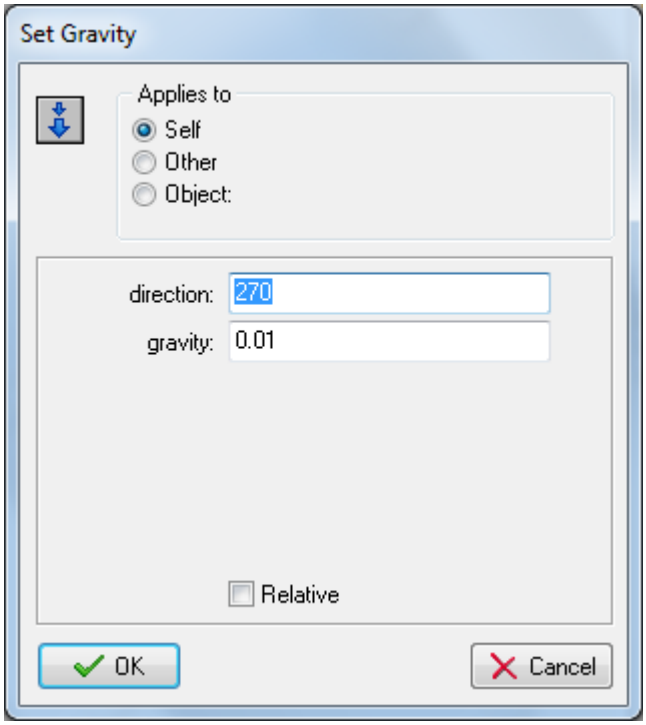
- Use this action to set the speed of an object.
- This is useful if you want an object to move one speed while travelling horizontally and another speed while travelling vertically.





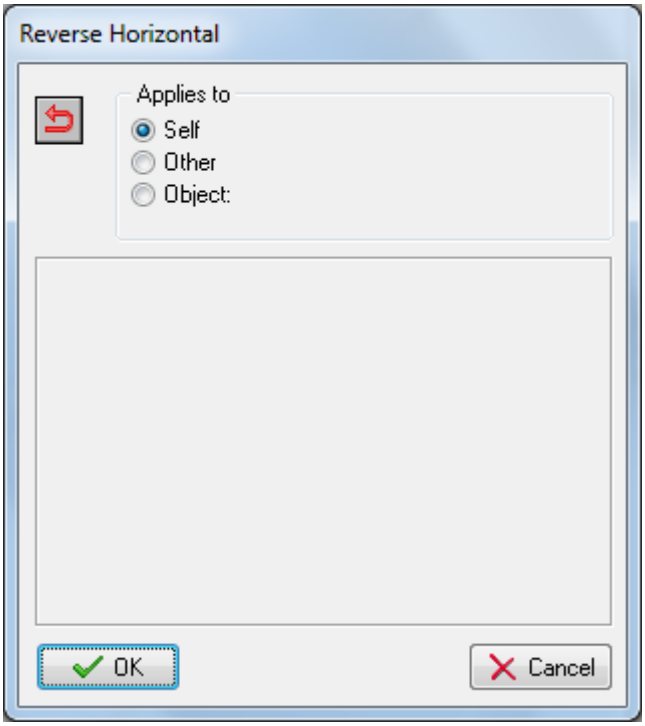
5. SET GRAVITY

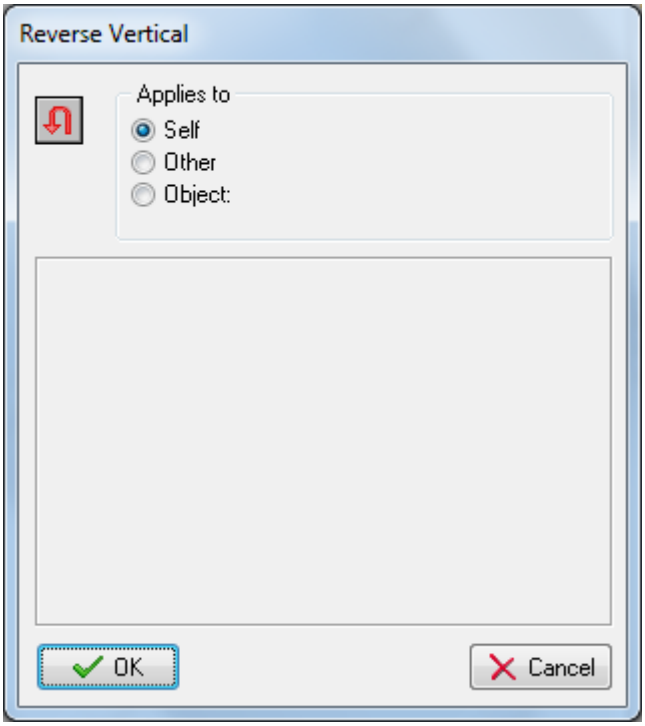
- You can set the direction you want the gravity to pull the object by using the degree of an angle (0 to 360 degrees).
- Typically you want a download direction (270 degrees).
- Normally you need a very small increment (for example, 0.01).
- The amount of speed in the given direction is added to the current motion of the object.
- Gravity is cumulative, meaning that an object will pick up speed, such as a boulder would do as it rolled down a mountainside.



6. REVERSE HORIZONTAL and VERTICAL

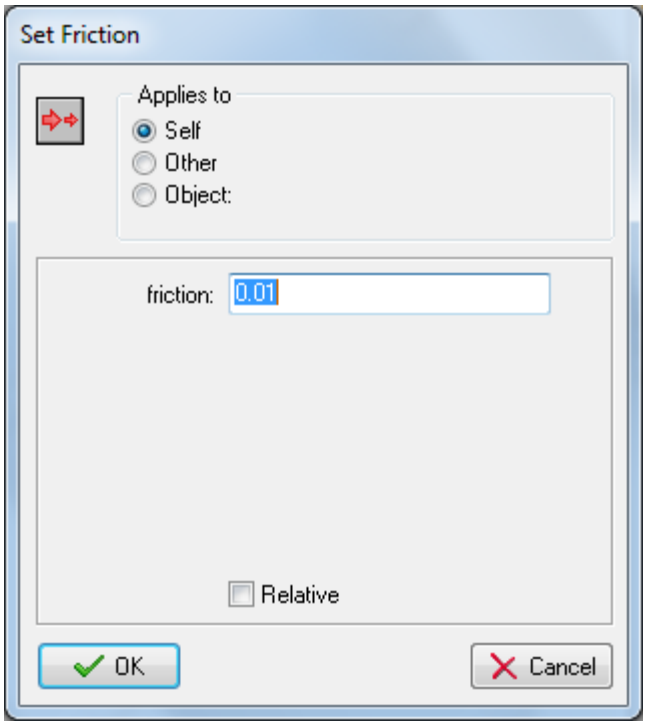
- This action is useful when an object collides with another object and you want the object to reverse direction.
- There are no values you need to set here; this function just reverses the horizontal and vertical direction of the object.





7. SET FRICTION

- This action slows down the object.
- For each step of the game, friction slows the object by the amount that you set.
- In each step, this amount is subtracted from the speed until the speed becomes 0.
- You may want to start with a small amount of friction, such as 0.01.



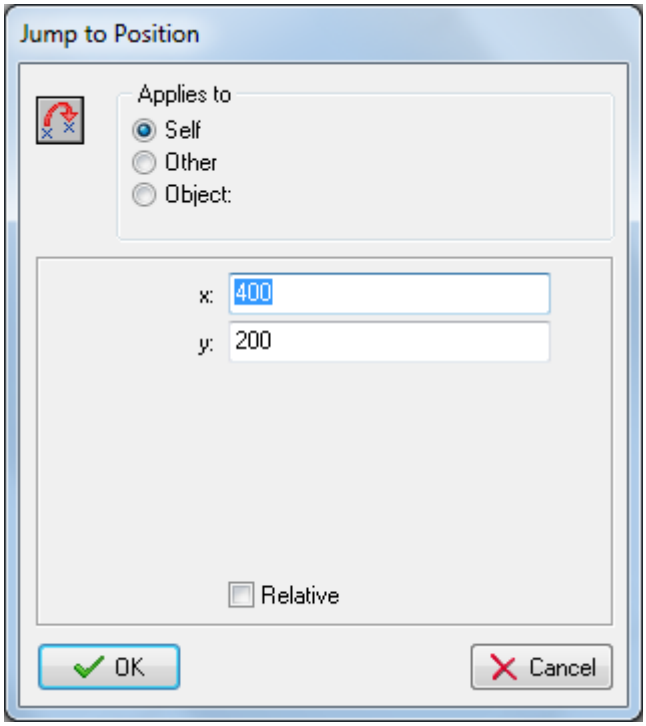
JUMP ACTIONS

Jump Actions allow you to move objects around a room. For example, at one moment an object may be at the bottom of the room, and in the next moment it could be at the top of the room.

The following are a list of the jump actions that are available:

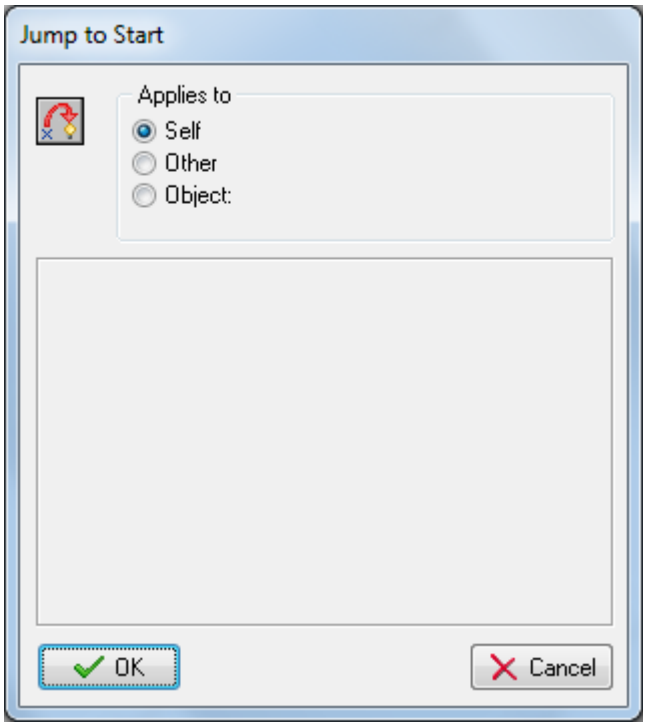
1. JUMP TO POSITION

- Using this action allows you to place an object in a particular position.
- You simply specify the x- and y-coordinate, and the object gets placed with its reference point on that position.
- If you check the **Relative** box, the position is relative to the current position of the object.
- This action is often used to continuously move an object.
- In each step you can increment the position a bit.



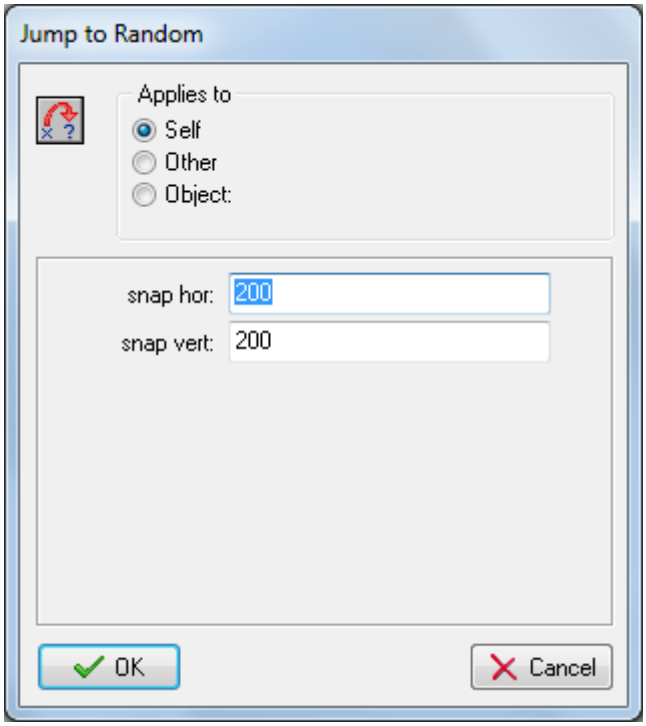
2. JUMP TO START

- Use this action to move an object back to where it was when it was first created.
- You don't need to specify anything for this action.



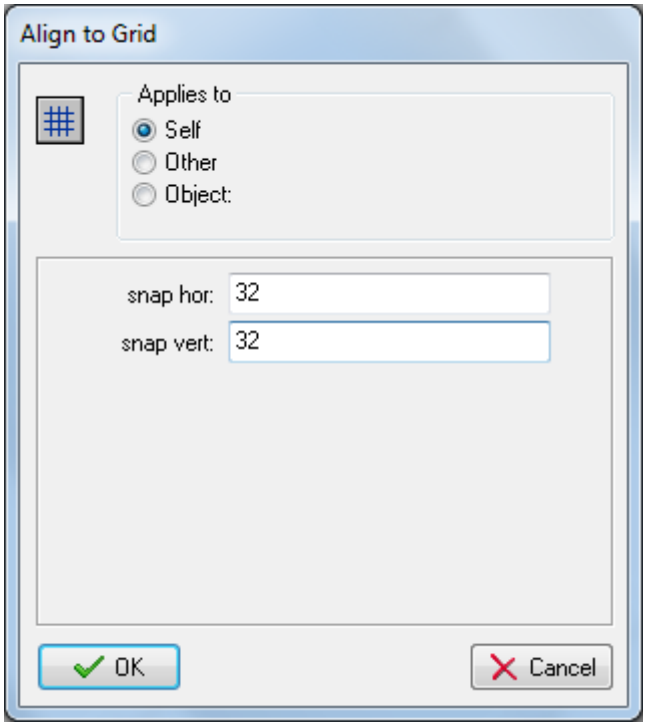
3. JUMP TO RANDOM

- Use this action to randomly move an object in a room.
- Objects won't appear where any other solid objects are located.
- You can specify the snapping used.
- If you specify positive values, the coordinates chosen will be integer multiples of the indicated values.
- This could for example be used to keep the instance aligned with the cells in your game (if any).
- You can specify a separate horizontal snapping and vertical snapping.



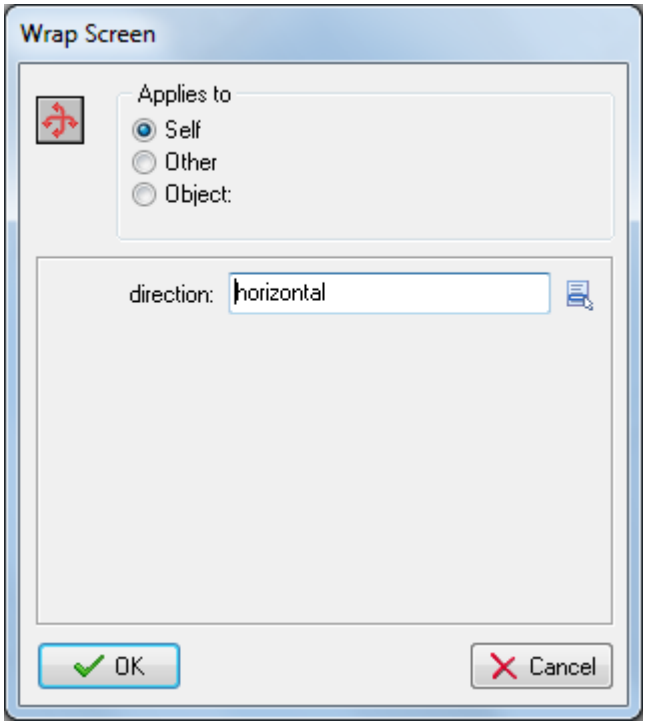
4. ALIGN TO GRID

- This action enables you to snap an object to a position for accurate placement within the game.
- You can indicate both the horizontal and vertical snapping value (that is, the size of the cells of the grid).
- This can be very useful to make sure that instances stay on a grid.



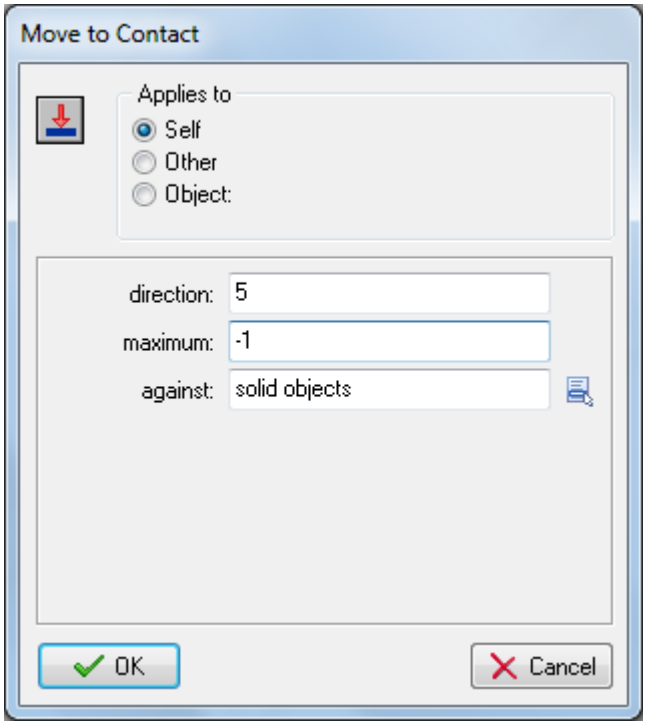
5. WRAP SCREEN

- Wrapping is when an object can leave one side of the screen and reappear on the other.
- This action is typically used together with the **Outside Room Event**; when the object is outside the room, wrap it around to the other side.
- You can choose **horizontal**, **vertical**, or **both**.



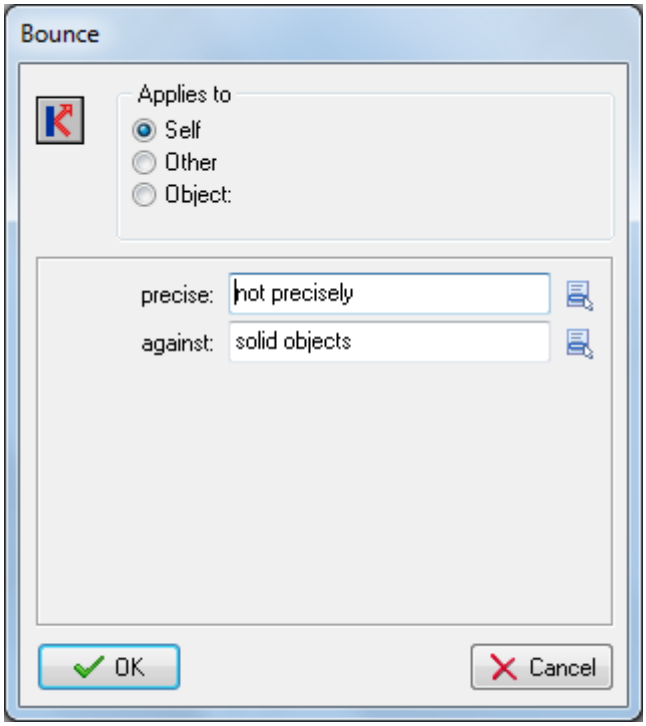
6. MOVE TO CONTACT

- With this action you can move an object in a given direction until a contact position with an object is reached.
- If there is already a collision at the current position the object is not moved. Otherwise, the object is placed just before a collision occurs.
- You can specify the direction as well as the maximum distance to move. For example, when the object is falling you can move a maximum distance down until an object is encountered.
- You can enter a value of –1 for the object to travel for an infinite distance (no maximum).
- You can also indicate whether to consider **solid objects** only or **all objects**.
- You typically put this action in the collision event to make sure that the object stops when it comes in contact with the other object involved in the collision.



7. BOUNCE

- Use this action when you want an object to naturally bounce off other objects.
- If you set **Precise** to **false**, you get the natural bounce effect only on straight walls; if you set **Precise** to **true**, you get the natural bounce even on slanted walls.
- You can also indicate whether to bounce only against **solid objects** or against **all objects**.
- The bounce is not completely accurate because it depends on many properties, but in many situations the effect is good enough.



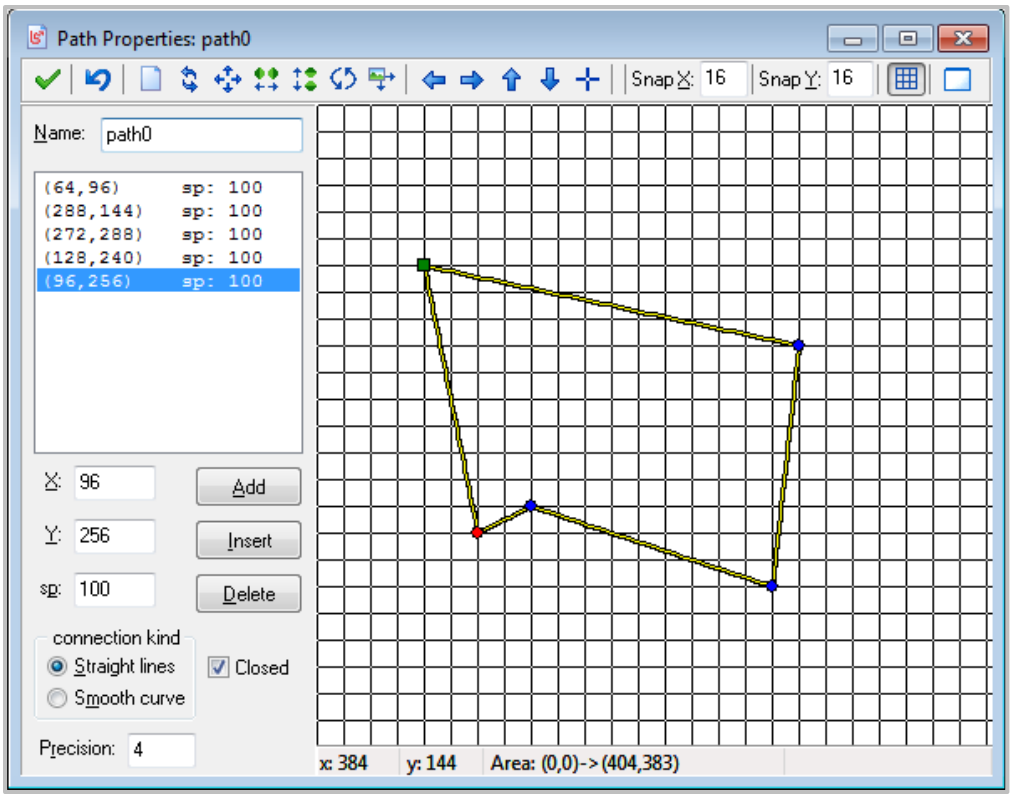
PATH ACTIONS

Paths are come in handy when you want to an object to travel in a specific way around a room. You can start a new path by clicking the **Path** icon from the menu or by right-clicking on **Paths** from the **Resource** tree.

Paths are represented as a yellow line. The points along the path are represented by colored squares and dots:

- The green square represents the starting and end point of the path.
- The red dot represents the selected way point.
- The blue dots represent way points.

You can create a path by clicking and dragging in the grid section of the **Path Properties**.

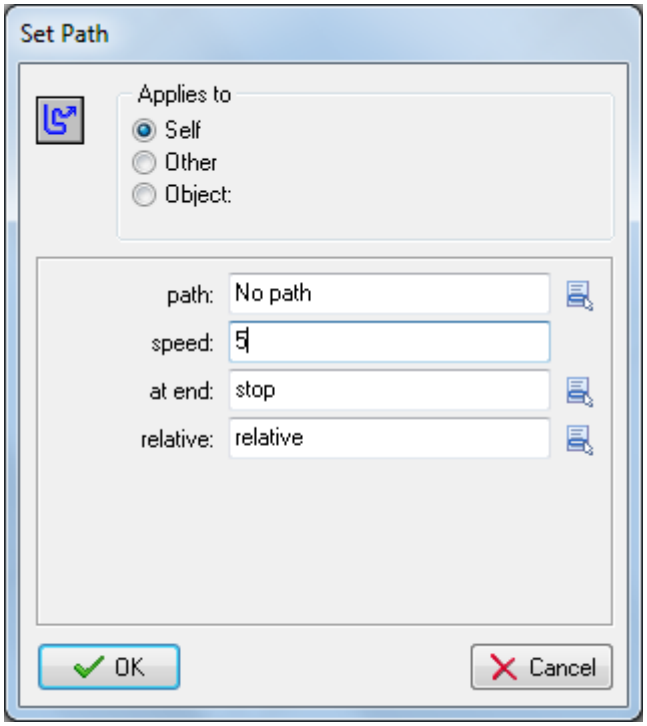


The **Path Actions** are configured directly to a path that you create and then assign to an action and an object.

The following are the **Path Actions** that are available:

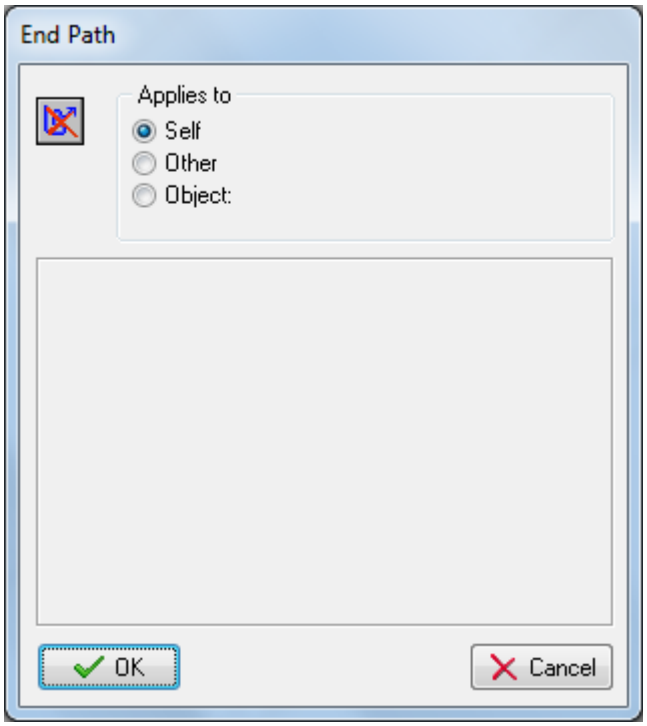
1. SET PATH

- With this action you can specify that the object should follow a particular path.
- You need to indicate the path that must be followed and the speed in pixels per step.
- When the speed is positive the instance starts at the beginning of the path.
- If it is negative it starts at the end.
- Next you specify the end behavior, that is, what should happen when the end of the path is reached. You can choose to **stop** the motion, **continue from start**, continue from here (which is restarting from the current position), or **reverse**.
- Finally you can indicate that the path must be seen as **absolute**, that is, the position will be as indicated in the path (this is useful when you have designed the path at a particular place in the room) or **relative**, in which case the start point of the path is placed at the current location of the object (end point when speed is negative).



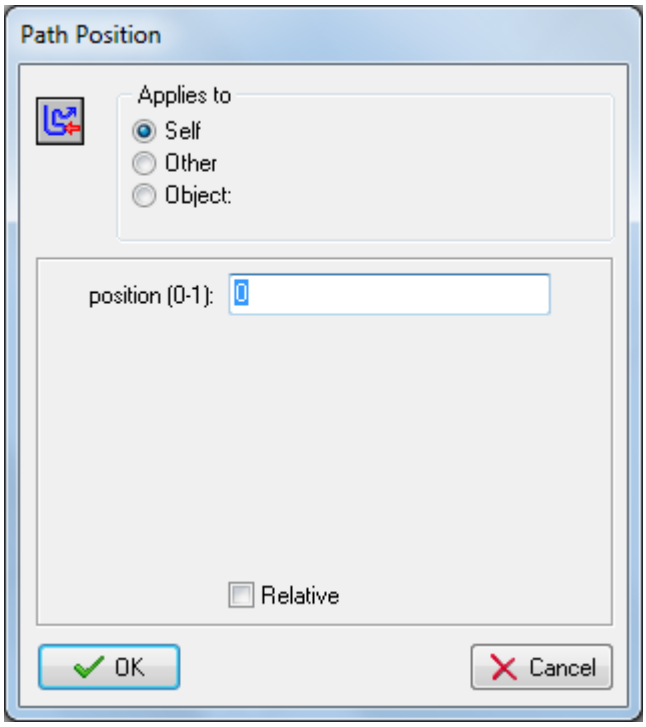
2. END PATH

- Use this action to stop the path for the object.



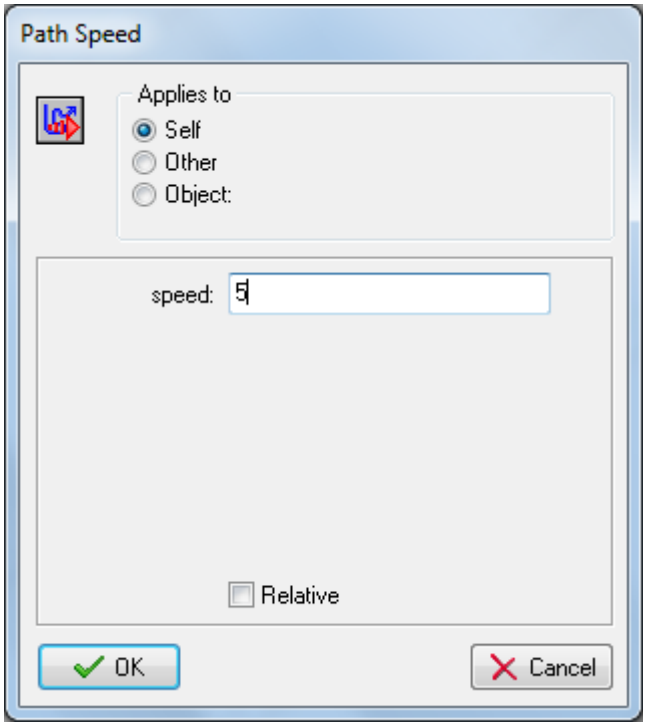
3. PATH POSITION

- Use this action to change the current position of the object in the path.
- You must use a value between 0 and 1, with 0 being the beginning and 1 being the end.



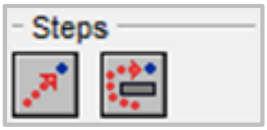
4. PATH SPEED

- With this action you can change the speed of the object on the path.
- A negative speed moves the object backwards along the path.
- Set it to 0 to temporarily stop the motion along the path.



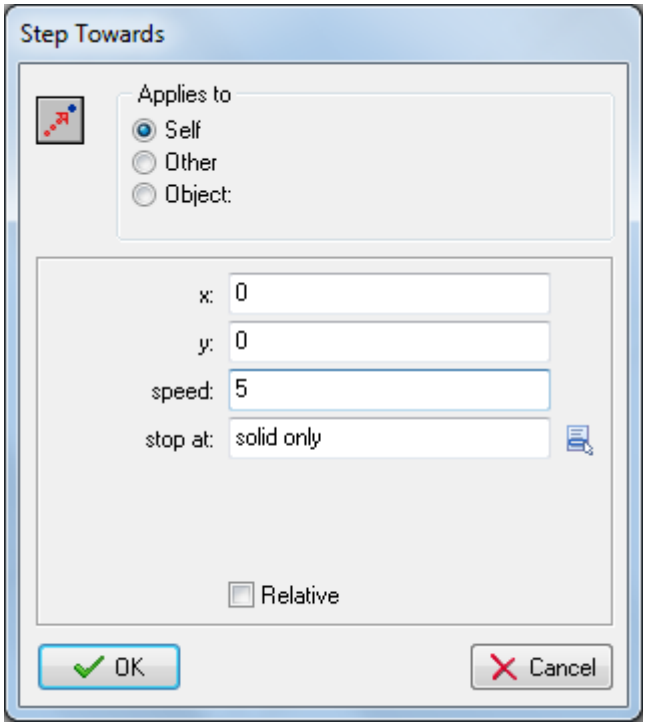
STEP ACTIONS

There are two **Step Actions** that you can use to step towards an object or to avoid an object.



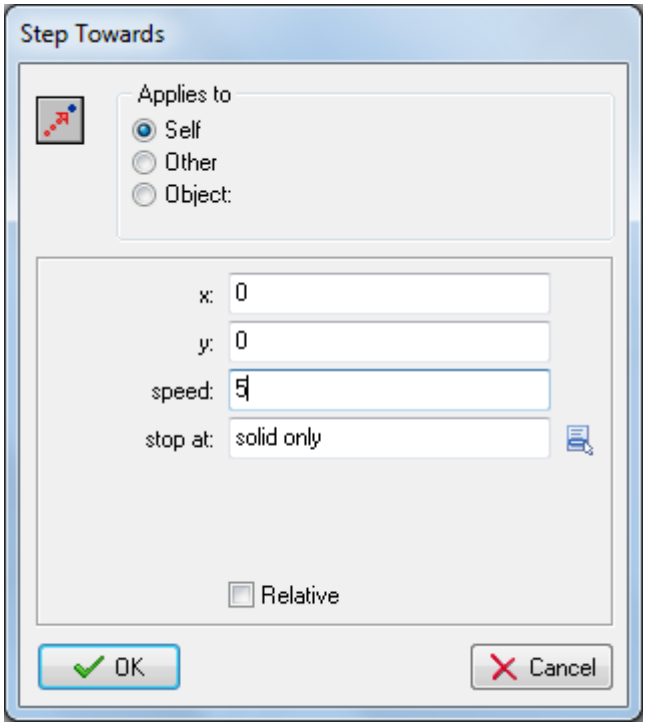
1. STEP TOWARDS

- Use this action to have an object move in a specific direction or toward another object.
- When the object is already at the position it will not move any further.
- You specify the position to move to, the speed with which to move (that is, the size of the step), and whether the motion should stop when hitting a solid instance or when hitting any instance.
- The difference between **Step Towards** and **Move Towards** is that in **Step Towards**, you can set a parameter for stopping the object when it encounters either solid objects or all objects.



2. STEP AVOID

- Like the **Step Towards** action, it lets the object take a step towards a particular position, but in this case it tries to avoid obstacles.
- When the object runs into a solid object (or any object) it will change the direction of motion to try to avoid the object and move around it.
- The approach is not guaranteed to work but in most easy cases it will effectively move the object towards the goal.
- You need to specify the position to move to, the speed with which to move (that is, the size of the step), and whether the motion should avoid solid objects or any objects.



RANDOMIZING VALUES

You may find it useful to set random values when moving, setting the speed, setting the location or setting the direction of an object. GameMaker includes a number of randomizing functions that allows you to randomly generate a value.

RANDOMLY GENERATING REAL NUMBERS

The **random** function returns a random real number. The syntax is **random(n)** where n is the upper range from which the random number will be selected. This function is good for probabilities where returning an integer (whole number) is not necessary.

For example, **random(100)** will return a value from 0 to 99, but that value can be 22.56473. You can also use real numbers and not integers in this function. For example, **random(0.5)** will return a value between 0 and 0.4999999.

RANDOMLY GENERATING INTEGERS

The **irandom()** function returns a random integer value. The syntax is **irandom(n)** where n is the upper range from which the random number will be selected.

So, for example, to get a random number from 0 to 9 you can use **irandom(9)** and it will return a number from 0 to 9 inclusive. Real numbers can also be used but the upper value will be excluded, so **irandom(9.7)** will return a value from 0 to 9 only.

RANDOMLY GENERATING FROM A RANGE

The **random_range()** function returns a random real number between the specified ranges. The syntax is **random_range(n1, n2)** where **n1** is the low end of the range from which the random number will be selected and **n2** is the upper end of the range from which the random number will be selected.

For example, **random_range(20,50)** will return a random number from 20 to 49, but the value may be a real number like 38.65265.

The **irandom_range()** function returns a random integer value between the specified ranges. The syntax is **irandom_range(n1, n2)** where **n1** is the low end of the range from which the random number will be selected and **n2** is the upper end of the range from which the random number will be selected.

For example, **irandom_range(10, 35)** will return an integer between 10 and 35. As with the **irandom()** function, real numbers can be used, in which case they will be rounded down to the nearest integer. For example, **irandom_range(6.2,9.9)** will give a value between 6 and 9.