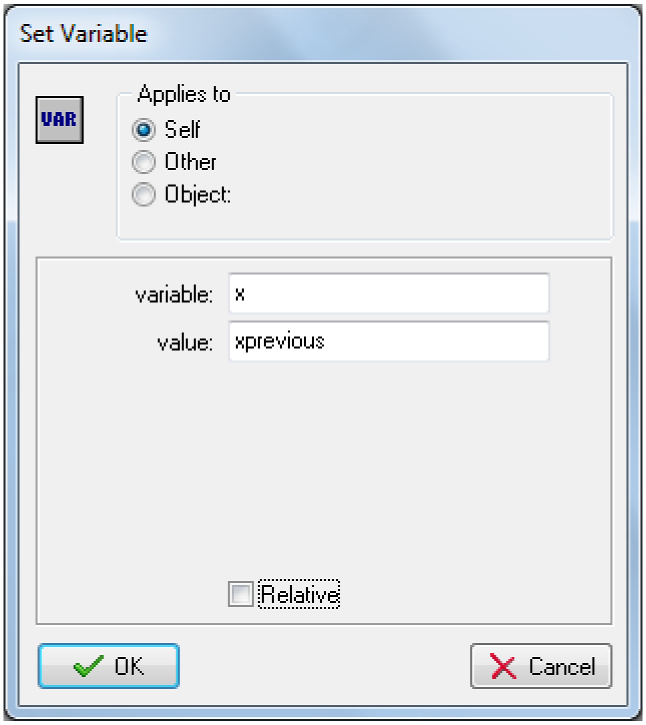
ADDING COLLISION EVENTS TO FISHPOD

It's time to start handling the remaining state transitions for our walking state object. This needs to include an event for handling (sideways) collisions with platforms, as well as a way of detecting whether the character is supported by a platform or not. We'll add the collision event first.

ADDING A COLLISION EVENT TO THE WALKING STATE OBJECT

1. Double-click the **obj_pod_walking** and add a **Collision** event with **obj_platform**. Remember that all the platform objects have **obj_platform** as their parent, so this collision event will be triggered for collisions between **obj_pod_walking** and any kind of platform.

2. Add a **Set Variable** action, type **x** into **Variable** and **xprevious** into **Value**. This sets the x position back to the value it had in the last step.



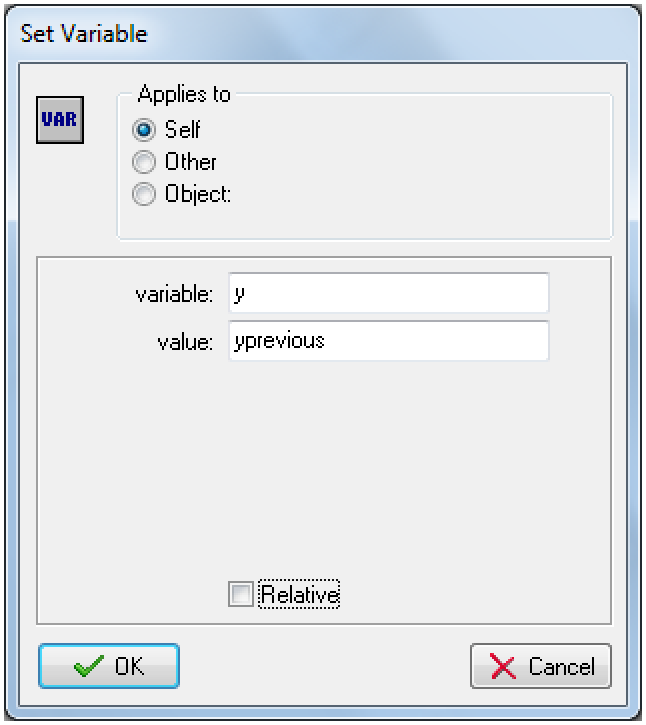
Set Variable

Applies to
● Self
○ Other
○ Object:

variable: x
value: xprevious

□ Relative

3. Add another **Set Variable** action, type **y** into **Variable** and **yprevious** into **Value**. This does the same for the y position.



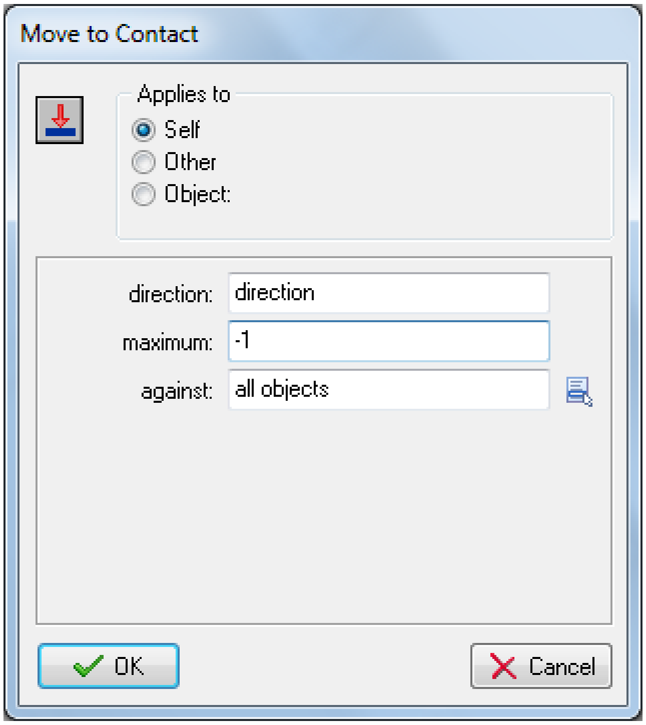
Applies to
● Self
○ Other
○ Object:

variable: y

value: yprevious

☐ Relative

4. Add a **Move to Contact** action, type **direction** into **Direction**, leave **Maximum** as **-1**, and select **all objects** for **Against**. This will find the contact point in the direction Fishpod is moving. The Maximum value indicates the distance (in pixels) that the action will check for a collision before giving up. A value of -1 means it will check for collisions an arbitrary distance away, but sometimes it is useful to be able to limit this.



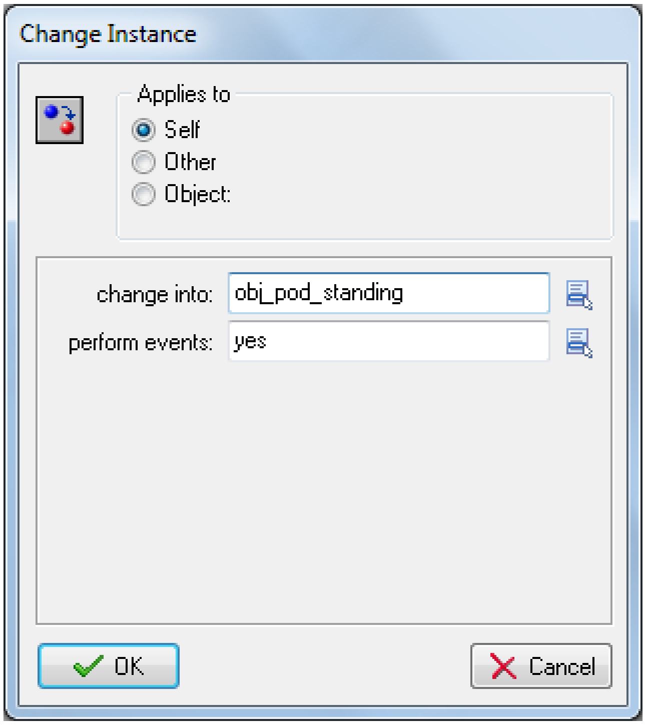
Applies to
● Self
○ Other
○ Object:

direction: direction

maximum: -1

against: all objects

5. Add a **Change Instance** action, select **obj_pod_standing** from the **Change Into** menu and **yes** for **Perform Events**. This will put the object into the standing state after a horizontal collision.



Applies to
● Self
○ Other
○ Object:

change into: obj_pod_standing

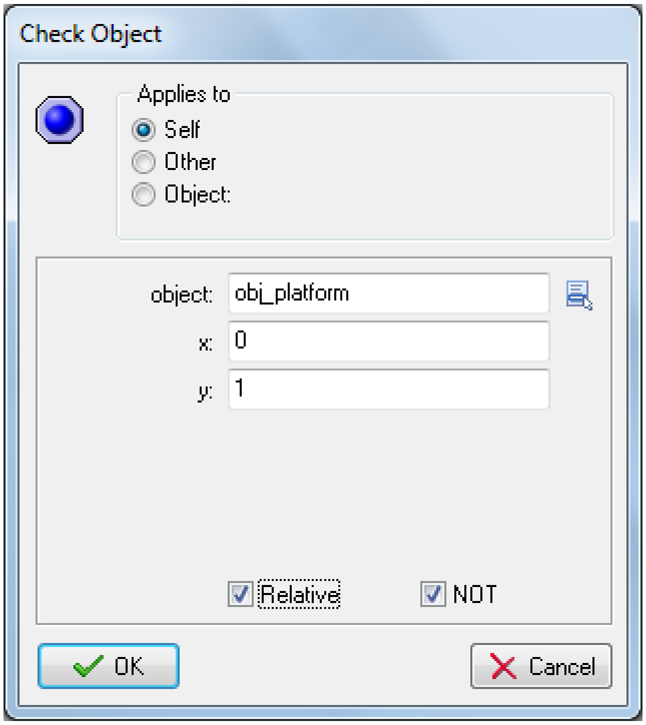
perform events: yes

Keep the walking state object open because we now need to handle the transition from the walking state to the falling state when there is no longer a platform supporting the character's feet. We can achieve this using the **Check Object** action.

ADDING A STEP EVENT TO THE WALKING STATE OBJECT

1. Add a **Step > Step Event** to **obj_pod_walking**. We're going to use a **Step** event because the walking object is constantly moving and the ground could stop supporting it at any point (a **Begin Step** or **End Step** would have an almost identical effect in this case).

2. Add a **Check Object** action, select **obj_platform** for **Object**, leave **X** as **0**, and set **Y** to **1**. Check both the **Relative** and **NOT** options. This checks to see if there would be a collision with an instance of **obj_platform** if Fishpod was moved down one pixel and only performs the next action if this is not true (that is, if Fishpod is standing in thin air).



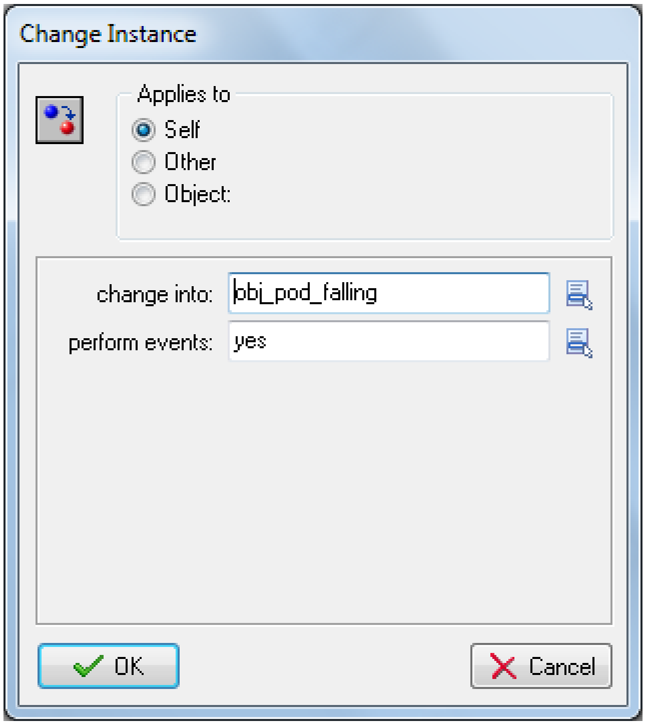
Check Object

Applies to
● Self
○ Other
○ Object:

object: obj_platform
x: 0
y: 1

☑ Relative ☑ NOT

3. Add a **Change Instance** action, select **obj_pod_falling** from the **Change Into** menu and **yes** for **Perform Events**. This will now put the object into the falling state when it is no longer supported by a platform.



Change Instance

Applies to
● Self
○ Other
○ Object:

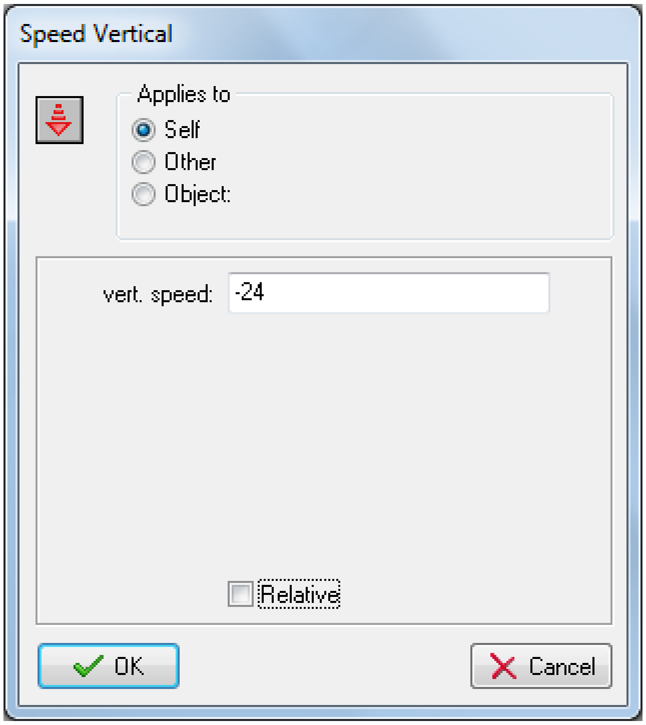
change into: obj_pod_falling
perform events: yes

This completes the walking state object, which now has all the necessary state transitions. Now we've just got the jumping and falling state objects to complete.

ADDING A CREATE EVENT FOR THE JUMPING STATE OBJECT

1. Double-click **obj_pod_jumping** and add a **Create** event.

2. Add a **Speed Vertical** action, type **-24** into **Vert. Speed**. A negative vertical speed moves the character up the screen, so this will launch Fishpod up into the air.



Speed Vertical

Applies to

● Self

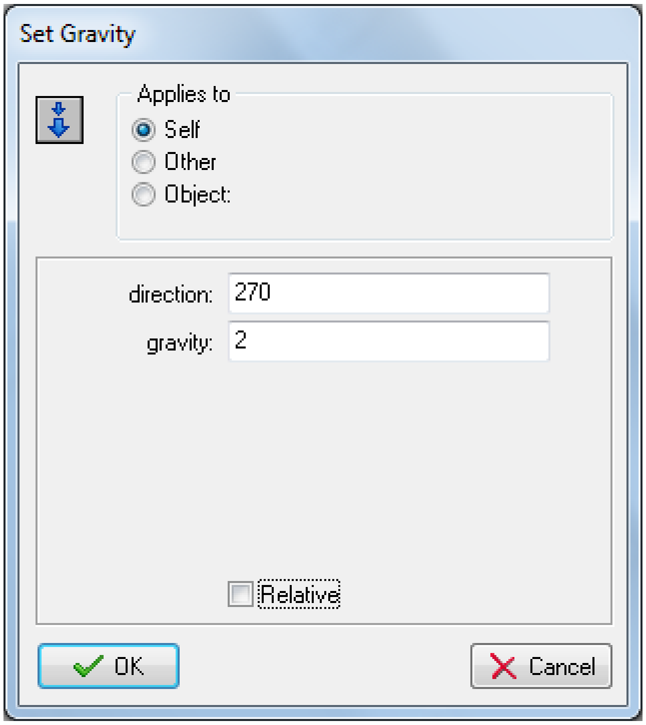
○ Other

○ Object:

vert. speed: -24

□ Relative

3. Add a **Set Gravity** action, type **270** into **Direction** (downward) and **2** into **Gravity**. This tells Game Maker to increase the vertical speed by **2** in every step, as if it was being pulled downward by the force of gravity.



Set Gravity

Applies to

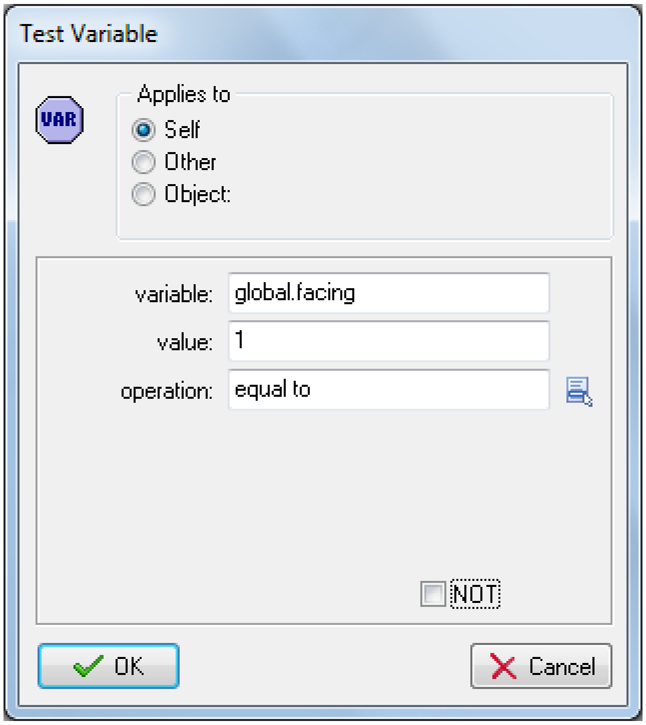
● Self
○ Other
○ Object:

direction: 270

gravity: 2

□ Relative

4. Add a **Test Variable** action, type **global.facing** into **Variable**, **1** into **Value**, and leave **Operation** set to **equal to**. This checks to see whether the character is facing left and only performs the next action(s) if this is true.



Test Variable

Applies to

● Self
○ Other
○ Object:

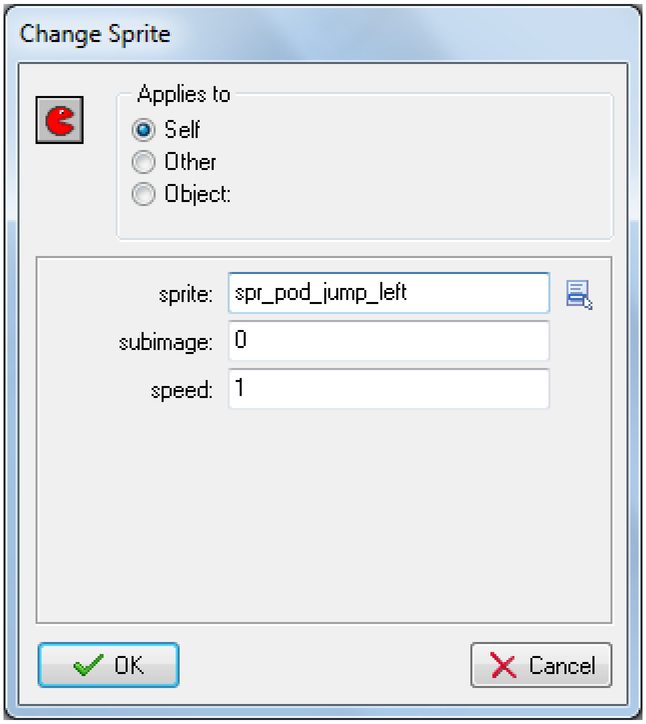
variable: global.facing

value: 1

operation: equal to

□ NOT

5. Add a **Start Block** action, add a **Change Sprite** action, and choose the **spr_pod_jump_left** sprite.



Change Sprite

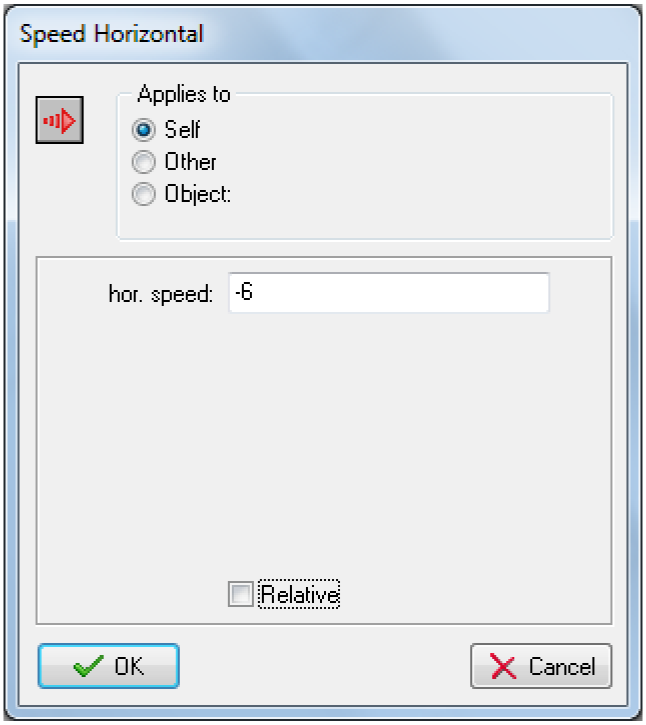
Applies to
● Self
○ Other
○ Object:

sprite: spr_pod_jump_left

subimage: 0

speed: 1

6. Add a **Speed Horizontal** action and set the **Hor. Speed** to **-6**.



Speed Horizontal

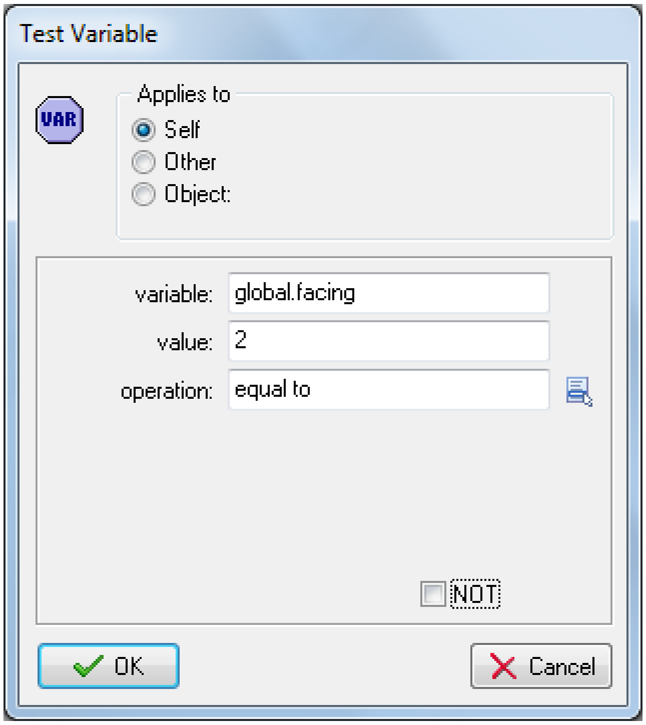
Applies to
● Self
○ Other
○ Object:

hor. speed: -6

□ Relative

7. Add an **End Block** action.

8. Repeat steps 4–7, testing for whether the character is facing right, changing the sprite to **spr_pod_jump_right**, and setting the **Hor. Speed** to **6**.



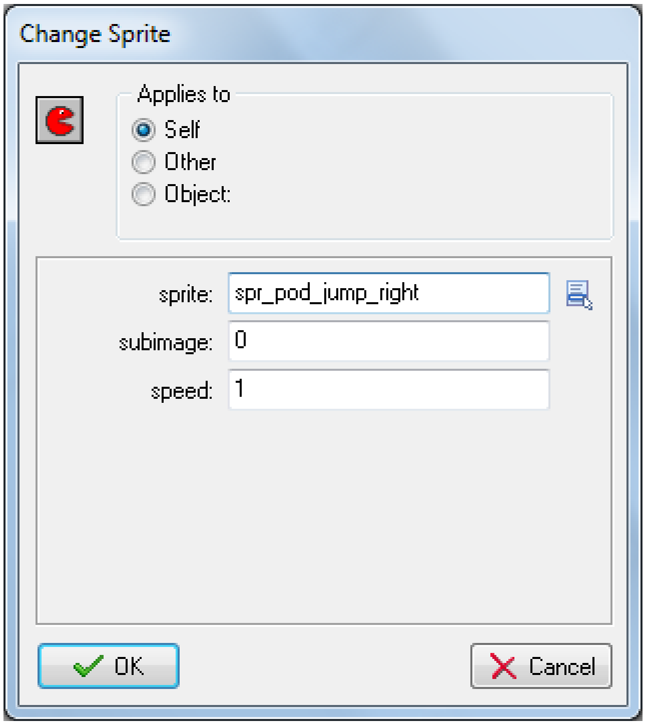
Test Variable

Applies to

● Self
○ Other
○ Object:

variable: global.facing
value: 2
operation: equal to

□ NOT

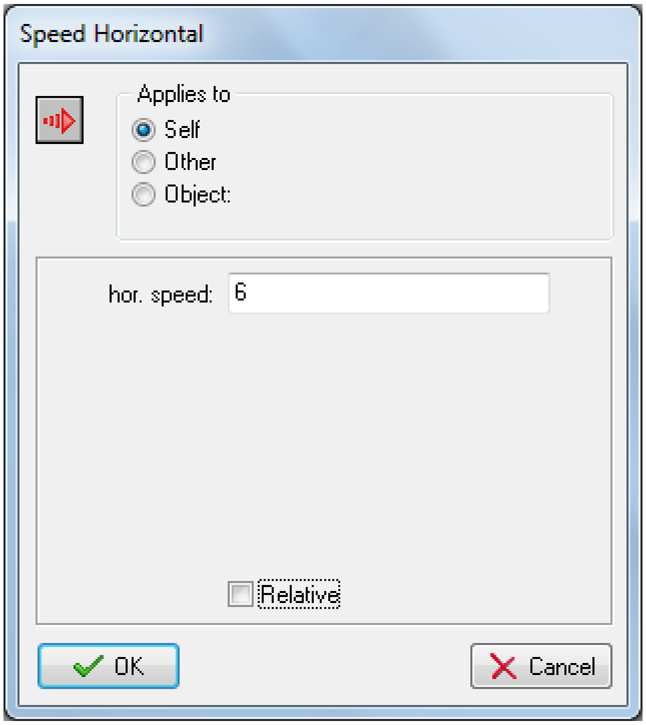


Change Sprite

Applies to

● Self
○ Other
○ Object:

sprite: spr_pod_jump_right
subimage: 0
speed: 1

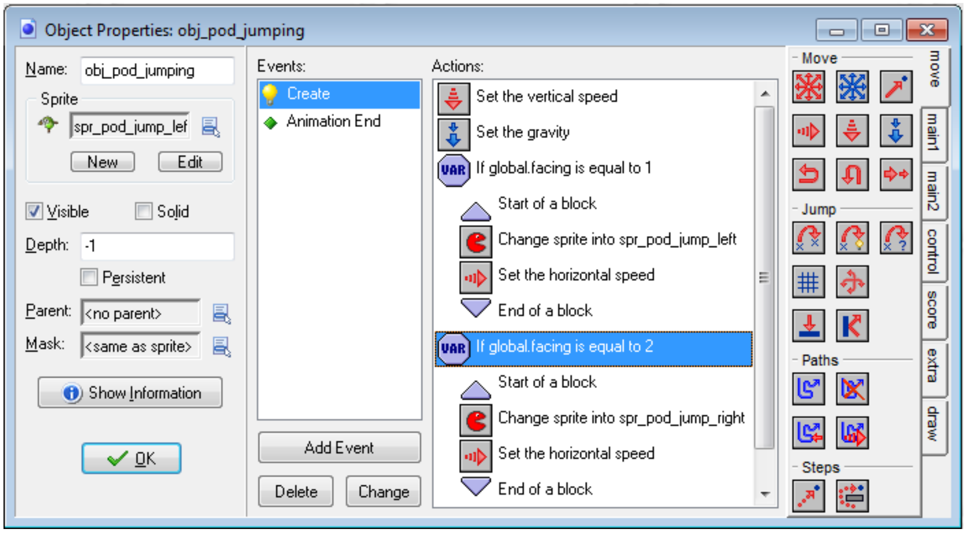


Applies to
● Self
○ Other
○ Object:

hor. speed: 6

□ Relative

Your **Create** event for **obj_pod_jumping** should look this this when you're done:



Name: obj_pod_jumping

Sprite: spr_pod_jump_lef

☑ Visible □ Solid

Depth: -1

□ Persistent

Parent: <no parent>

Mask: <same as sprite>

Events:
- Create
- Animation End

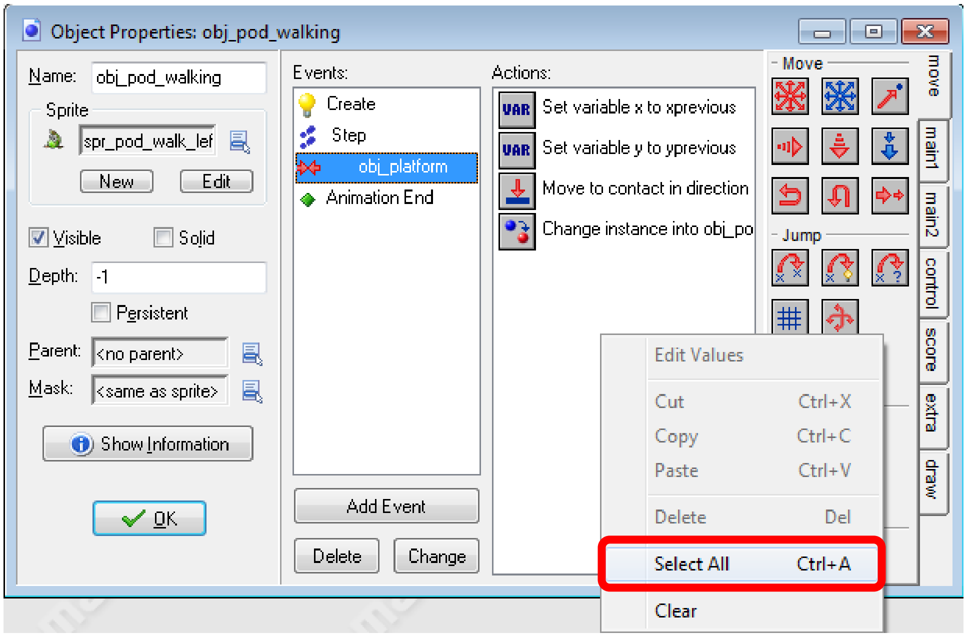
Actions:
- Set the vertical speed
- Set the gravity
- If global.facing is equal to 1
  - Start of a block
  - Change sprite into spr_pod_jump_left
  - Set the horizontal speed
  - End of a block
- If global.facing is equal to 2
  - Start of a block
  - Change sprite into spr_pod_jump_right
  - Set the horizontal speed
  - End of a block

COPYING A COLLISION EVENT FOR THE JUMPING STATE OBJECT

Our collision event for the jumping state object is actually going to be very similar to the collision event for the walking object, so we'll copy it to save some time. Remember that we used the **direction** variable as a parameter to the **Move to Contact** action, so the actions we're copying automatically handle collisions in whatever direction Fishpod is travelling. However, we'll need to change the **Change Instance** action from **obj_pod_standing** to **obj_pod_falling** and we'll require an additional action in there that brings the jumping Fishpod to a halt.

1. Double-click **obj_pod_walking** and select the **Collision** event with **obj_platform**.

2. Right-click anywhere in the actions list and choose **Select All** from the menu.



Object Properties: obj_pod_walking

Name: obj_pod_walking

Sprite: spr_pod_walk_lef

☑ Visible □ Solid

Depth: -1

□ Persistent

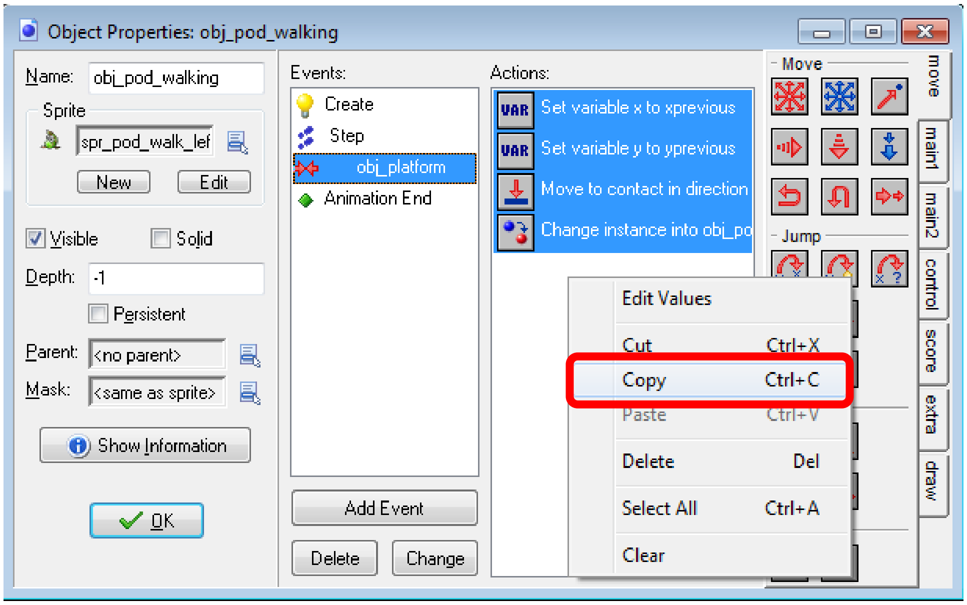
Parent: <no parent>

Mask: <same as sprite>

Events:
- Create
- Step
- obj_platform
- Animation End

Actions:
- Set variable x to xprevious
- Set variable y to yprevious
- Move to contact in direction
- Change instance into obj_po

3. Right click again and choose **Copy**.



Object Properties: obj_pod_walking

Name: obj_pod_walking

Sprite: spr_pod_walk_lef

☑ Visible □ Solid

Depth: -1

□ Persistent

Parent: <no parent>

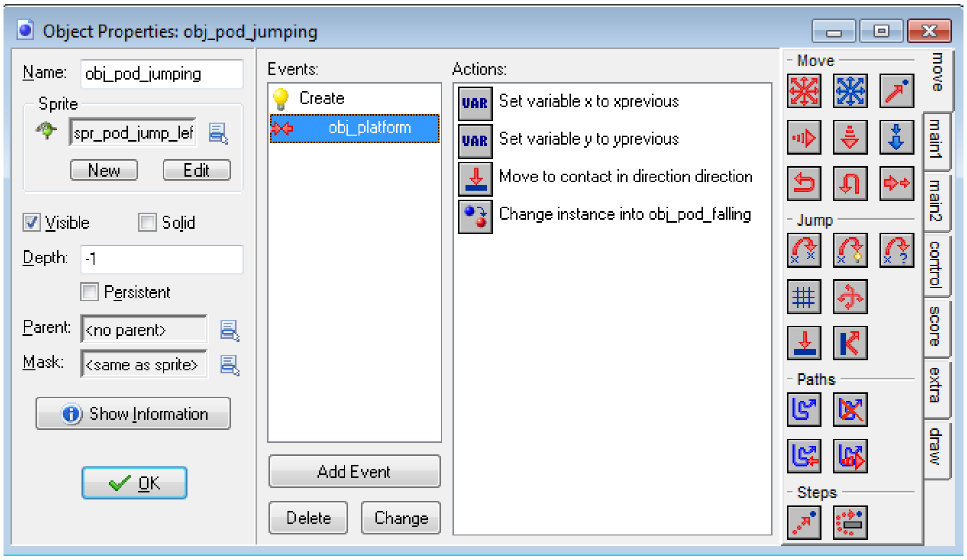
Mask: <same as sprite>

Events:
- Create
- Step
- obj_platform
- Animation End

Actions:
- Set variable x to xprevious
- Set variable y to yprevious
- Move to contact in direction
- Change instance into obj_po

4. Double-click **obj_pod_jumping** and add a **Collision** event with **obj_platform**.

5. Right-click anywhere in the actions list and choose **Paste** from the menu. All four actions from **obj_pod_walking** should be copied over.


Object Properties: obj_pod_jumping

Name: obj_pod_jumping

Sprite: spr_pod_jump_lef

☑ Visible □ Solid

Depth: -1

□ Persistent

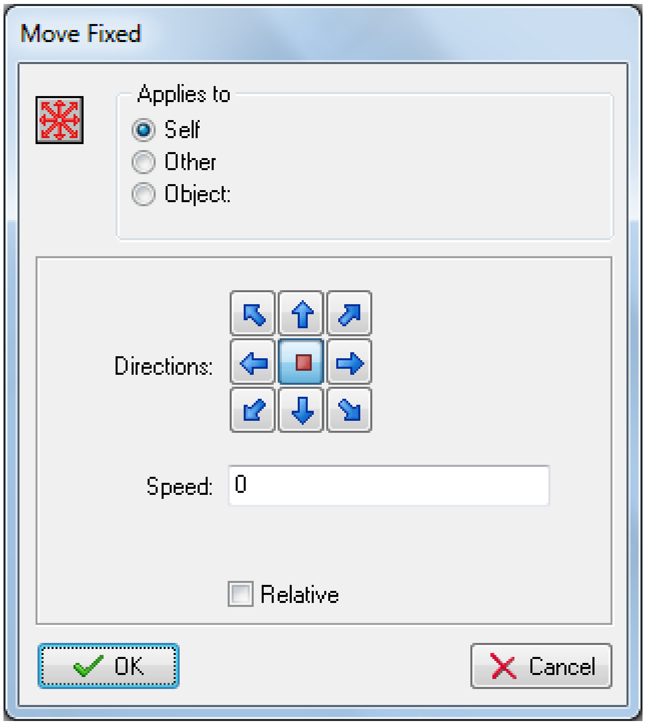
Parent: <no parent>

Mask: <same as sprite>

Events:
- Create
- obj_platform

Actions:
- Set variable x to xprevious
- Set variable y to yprevious
- Move to contact in direction direction
- Change instance into obj_pod_falling

6. Add the **Move Fixed** action into the actions list before the Change Instance action, select the **middle** direction button to indicate no direction of moment and leave the **Speed** set to **0**.


Move Fixed

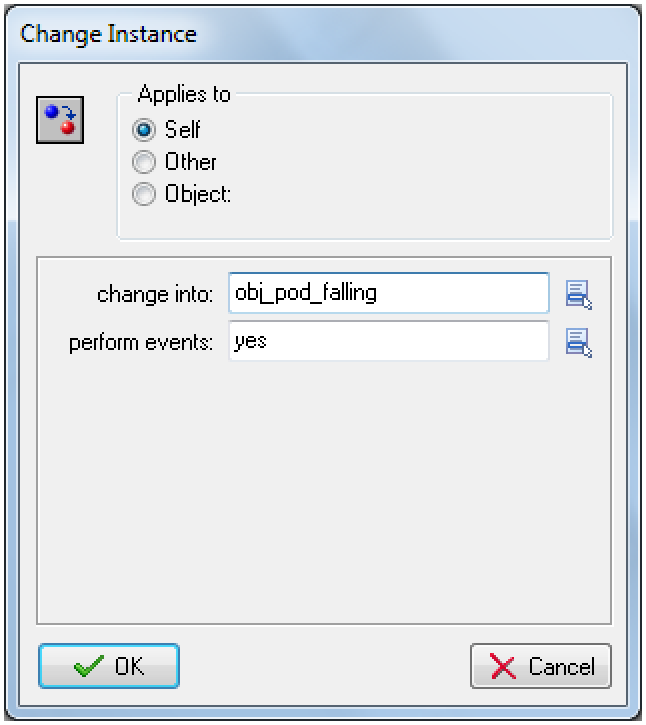
Applies to
- ● Self
- ○ Other
- ○ Object:

Directions:

Speed: 0

□ Relative

7. Double-click the **Change Instance** action in the actions list and change the **Change Into** option to **obj_pod_falling**.



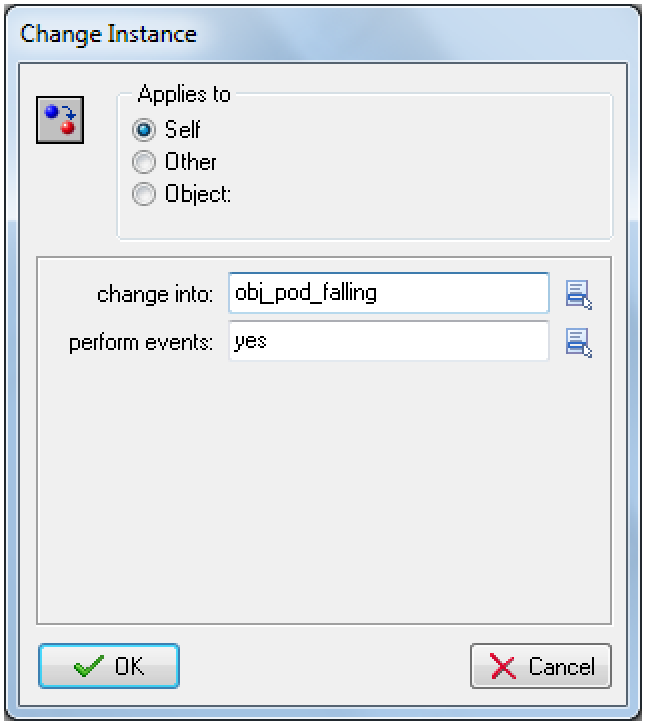
change into: obj_pod_falling

perform events: yes

ADDING AN ANIMATION END EVENT FOR THE JUMPING STATE OBJECT

Finally we will handle the **Animation End** event, which just needs to switch from the rotating jumping animation into the static falling animation at the end of one full rotation.

1. Add an **Other > Animation End** event to **obj_pod_jumping**.
2. Add a **Change Instance** action that changes into **obj_pod_falling** and set **Perform Events** to **yes**.



change into: obj_pod_falling

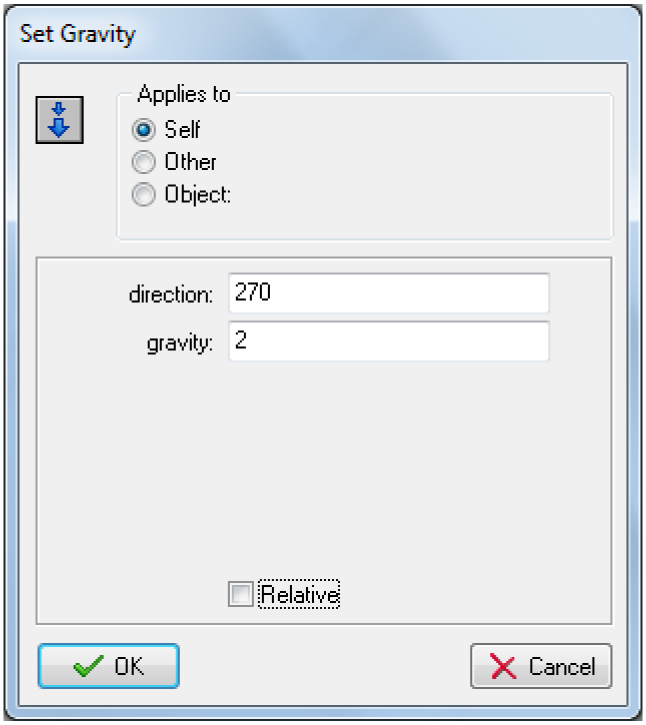
perform events: yes

ADDING A CREATE EVENT TO THE FALLING STATE OBJECT

The only missing state behavior is for the falling state object. It's not particularly complicated, as it just needs to be affected by gravity. However, the **Set Gravity** action in Game Maker continues to increase the speed of an object indefinitely, so we must be careful not to break our collision detection by making the object move too fast.

Let's start with the **Create** event to set up the default object behavior.

1. Double-click the **obj_pod_falling** object and add a **Create** event.
2. Add a **Set Gravity** action, set **Direction** to **270** (downwards) and **Gravity** to **2**.



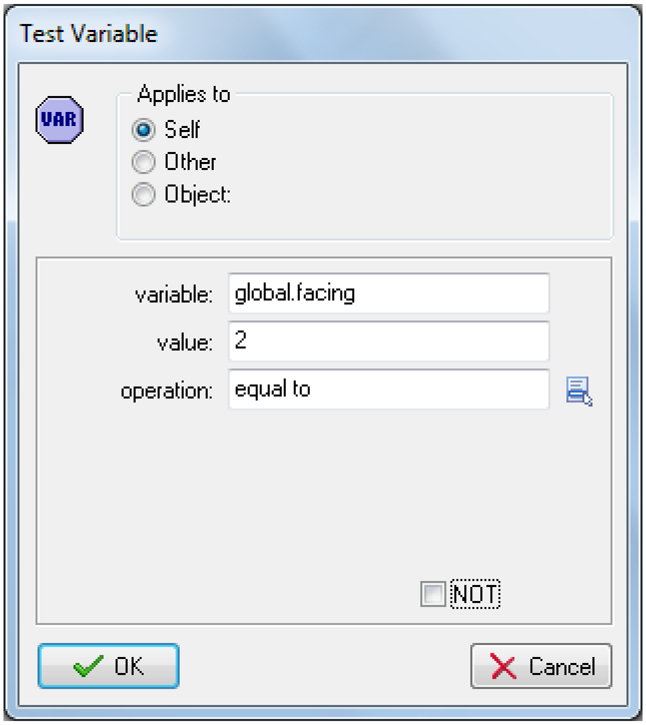
Applies to
● Self
○ Other
○ Object:

direction: 270

gravity: 2

□ Relative

3. Add a **Test Variable** action and check that **global.facing** is equal to **2.**



Applies to
● Self
○ Other
○ Object:

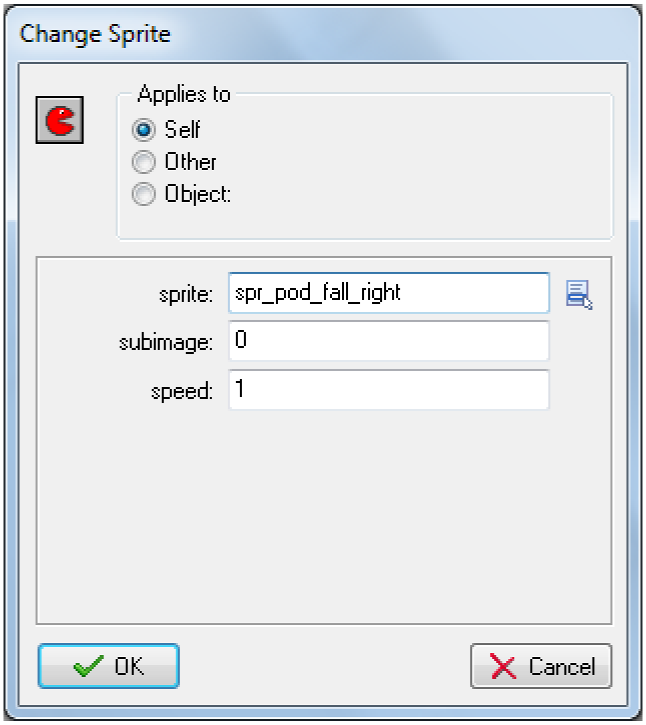
variable: global.facing

value: 2

operation: equal to

□ NOT

4. Add a **Change Sprite** action and select **spr_pod_fall_right**.

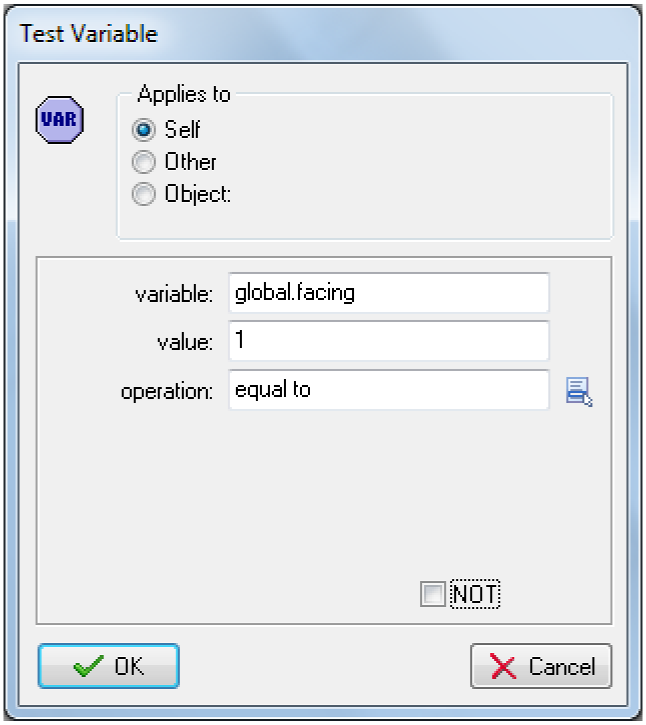


Change Sprite

Applies to
● Self
○ Other
○ Object:

sprite: spr_pod_fall_right
subimage: 0
speed: 1

5. Add another **Test Variable** action and check that **global.facing** is equal to **1**.



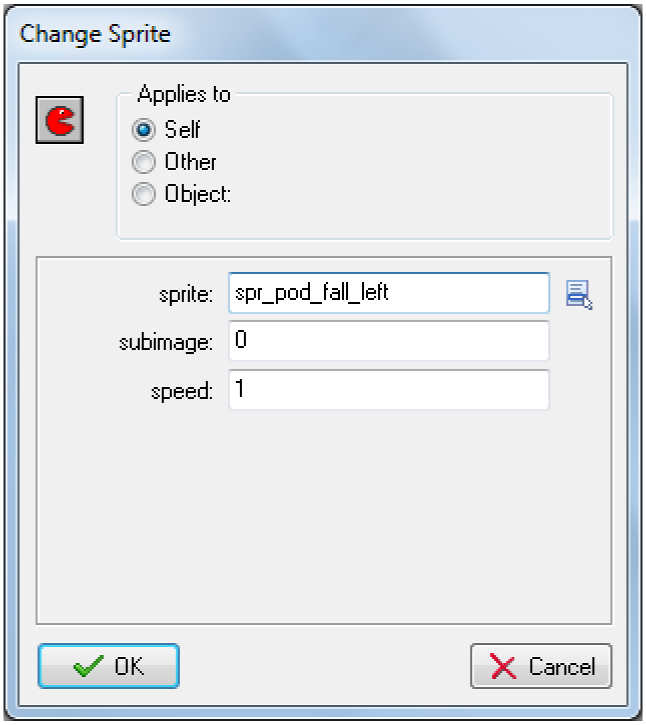
Test Variable

Applies to
● Self
○ Other
○ Object:

variable: global.facing
value: 1
operation: equal to

□ NOT

6. Add a **Change Sprite** action and select **spr_pod_fall_left**.



Change Sprite

Applies to

● Self
○ Other
○ Object:

sprite: spr_pod_fall_left

subimage: 0

speed: 1

COPYING A COLLISION EVENT FOR THE FALLING STATE OBJECT

Next, we will copy the collision event from the jumping state object, and implement the transition back into the standing state when Fishpod is supported by a platform. This collision needs to check whether there is actually a platform beneath Fishpod's feet, as he may have fallen diagonally into the side of a platform after making a jump.

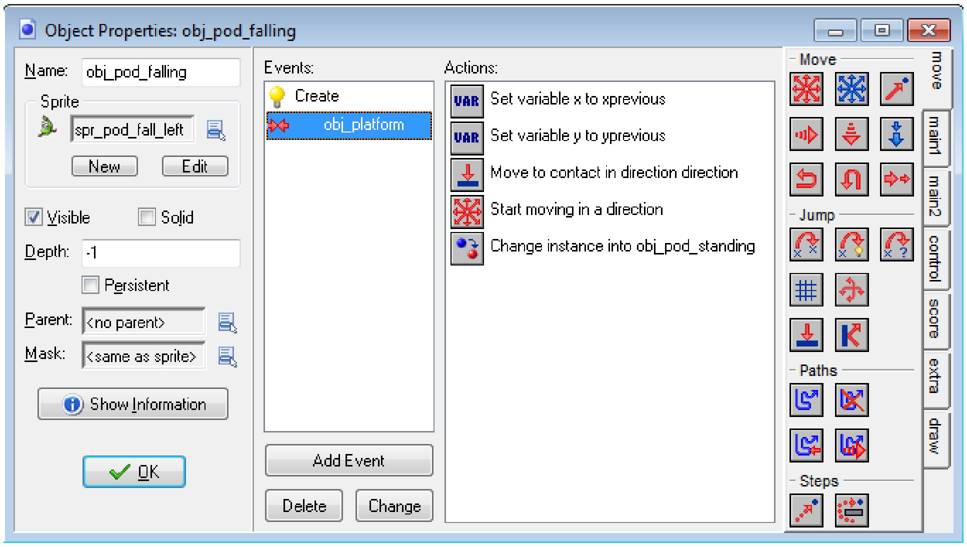
1. Double-click **obj_pod_jumping** and select the **Collision** event with **obj_platform**.

2. Right-click anywhere in the actions list and choose **Select All** from the menu.

3. Right-click again and choose **Copy**.

4. Reopen **obj_pod_falling** and add a **Collision** event with **obj_platform**.

5. Right-click anywhere in the actions list and choose **Paste** from the menu. All five actions from **obj_pod_jumping** should be copied over.


Object Properties: obj_pod_falling

Name: obj_pod_falling

Sprite: spr_pod_fall_left

☑ Visible ☐ Solid

Depth: -1

☐ Persistent

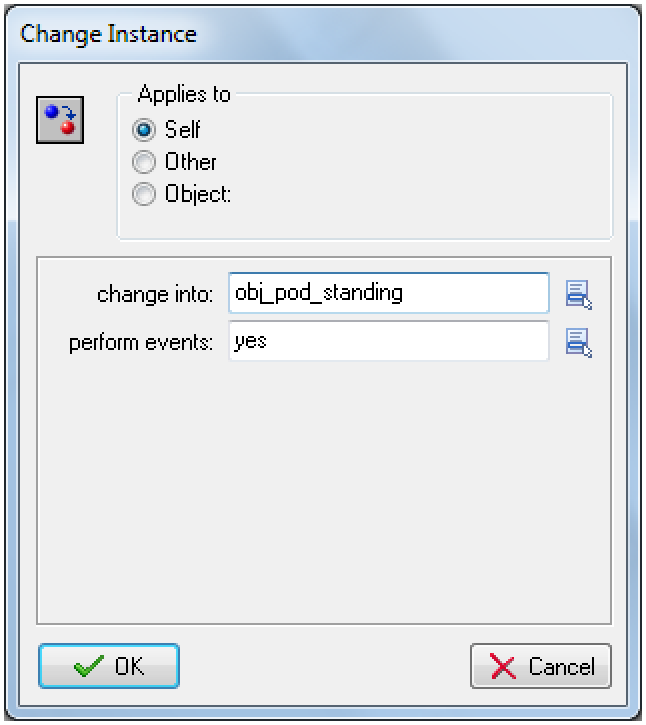
Parent: <no parent>

Mask: <same as sprite>

Events:
- Create
- obj_platform

Actions:
- Set variable x to xprevious
- Set variable y to yprevious
- Move to contact in direction direction
- Start moving in a direction
- Change instance into obj_pod_standing

6. Double-click the **Change Instance** action and change the **Change Into** option to **obj_pod_standing** and set **Perform Events** to **yes**.

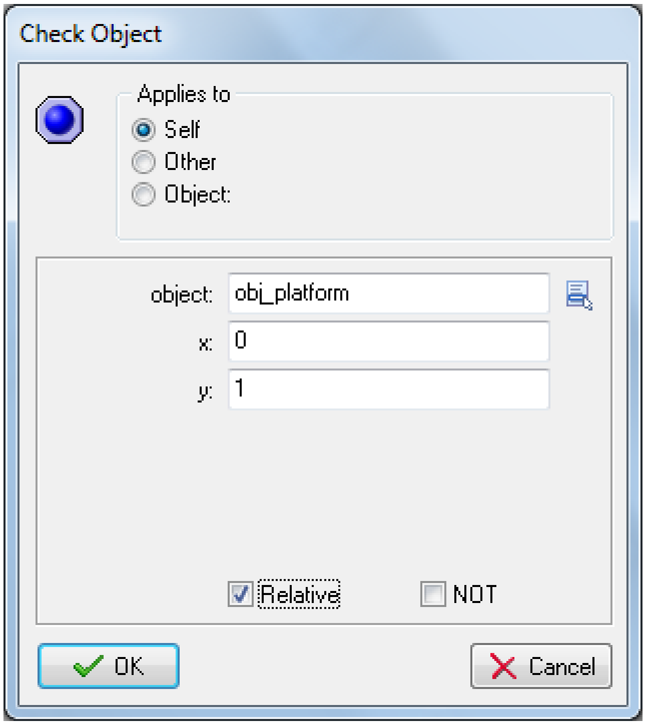

Change Instance

Applies to
- ● Self
- ○ Other
- ○ Object:

change into: obj_pod_standing

perform events: yes

7. Add a **Check Object** action directly above the **Change Instance** action, set **Object** to **obj_platform**, set **Y** to **1**, and check the **Relative** option. This will now only switch to the standing state if Fishpod is supported after colliding with the platform. This is important in the situation where Fishpod falls sideways into a platform as it doesn't make sense for him to change state when that happens.

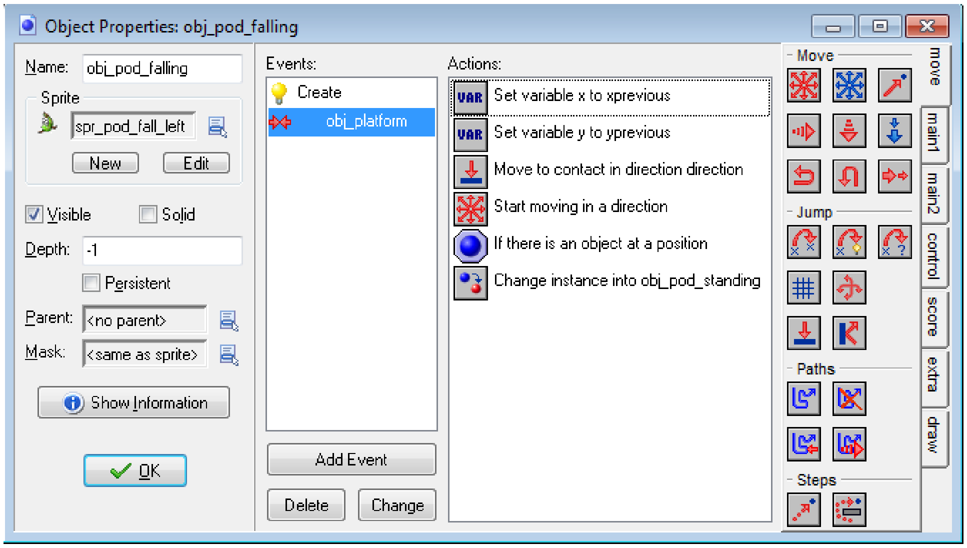

Check Object

Applies to
● Self
○ Other
○ Object:

object: obj_platform
x: 0
y: 1

☑ Relative ☐ NOT

Your **Collision** event for **obj_pod_falling** should look like this when you're done:


Object Properties: obj_pod_falling

Name: obj_pod_falling
Sprite: spr_pod_fall_left
☑ Visible ☐ Solid
Depth: -1
☐ Persistent
Parent: <no parent>
Mask: <same as sprite>

Events:
- Create
- obj_platform

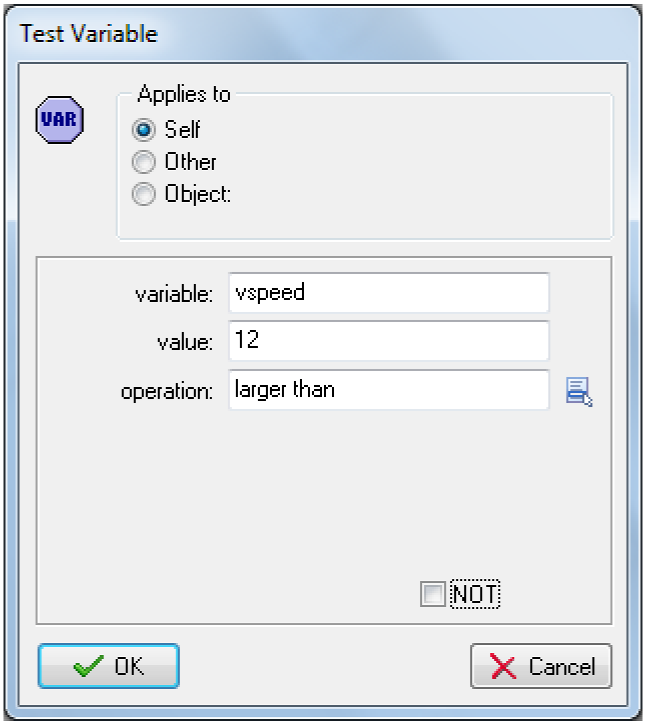
Actions:
- Set variable x to xprevious
- Set variable y to yprevious
- Move to contact in direction direction
- Start moving in a direction
- If there is an object at a position
- Change instance into obj_pod_standing

ADDING A STEP EVENT TO THE FALLING STATE OBJECT

Now we need to handle the problem of gravity making Fishpod fall too fast for the collision detection. We'll do this in a **Step** event because this will constantly keep the speed in check.

1. Add a **Step > Step Event** to the **obj_pod_falling** object.

2. Add a **Test Variable** action and use it to check that **vspeed** (the object's vertical speed) is larger than 12.

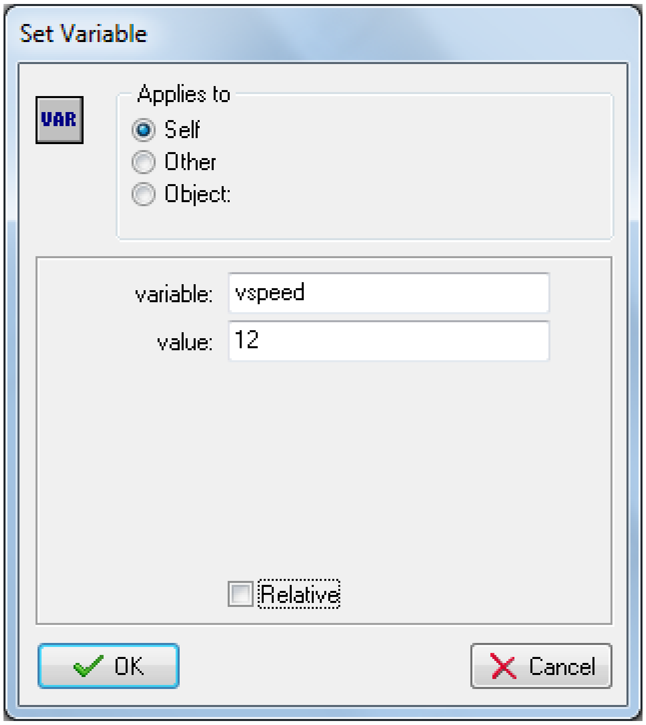


Test Variable

Applies to
● Self
○ Other
○ Object:

variable: vspeed
value: 12
operation: larger than
□ NOT

3. Add a **Set Variable** action and use it to set **vspeed** to **12**. This is equivalent to using a **Speed Vertical** action, which sets the **Vert. Speed** to **12**. This will ensure that the vertical speed of Fishpod never exceeds 12 pixels in each step. Its bounding box is 36 pixels high, so there should never be any gaps in the collision detection when it is falling.



Set Variable

Applies to
● Self
○ Other
○ Object:

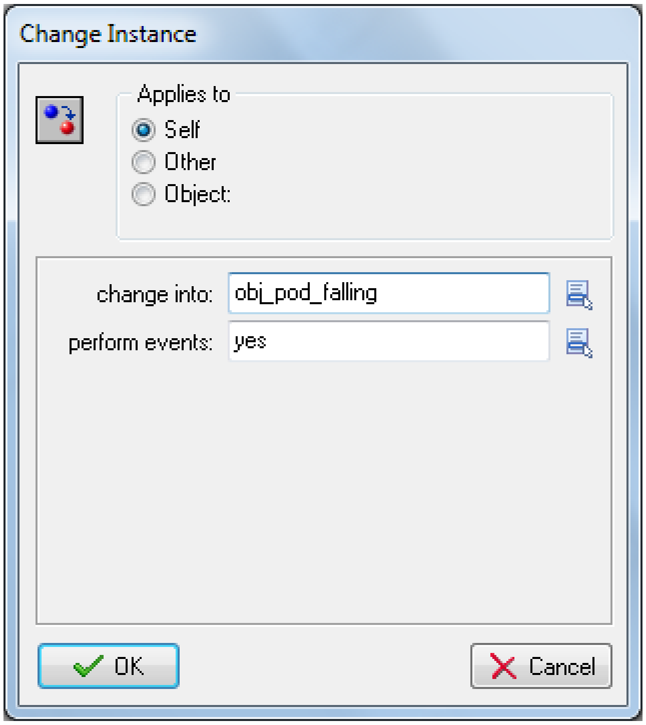
variable: vspeed
value: 12
□ Relative

EDITING THE CREATE EVENT OF THE FISHPOD OBJECT

That's just about it for Fishpod's state objects. There is just one more change we should make to **obj_pod** before running the game. We originally set the initial state of Fishpod to be **obj_pod_standing**, as this was the only state we had implemented at that stage. It makes more sense now to start Fishpod in the falling state, as this handles the situation where he is not supported by a platform at the start of the level.

1. Double-click the **obj_pod** object and select the **Create** event.

2. Edit the **Change Instance** action and set it to **obj_pod_falling**.



Applies to
● Self
○ Other
○ Object:

change into: obj_pod_falling
perform events: yes

Now when you run the game, it should start to feel like a platform game. You should be able to walk along platforms and jump between them. Try not to jump off the screen at the moment, as we've not implemented a way to restart the level without stopping the game yet.

SOURCE: Habgood, Jacob, Nielsen, Nana, Rijks, Martin and Kevin Crossley. *The Game Maker's Companion: Game Development: The Journey Continues.* New York: Apress, 2010. Print.