1 Platformer Guide

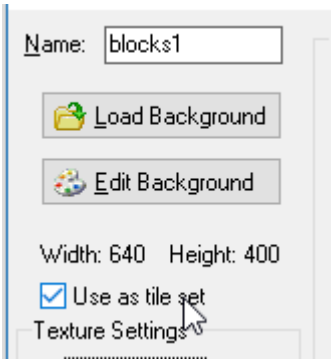
2 SETUP

Unzip the **Platformer.zip** file to a convenient location.

Start a new Gamemaker project. Name it **Platformer**.

NOTE: THE INSTRUCTIONS IN THIS GUIDE ASSUME YOU HAVE BUILT SHOOTINGGALLERY, SPACERESCUE AND SPACESHOOTER. THEY RAPIDLY STEP THROUGH TECHNIQUES AND IDEAS YOU WERE EXPOSED TO PREVIOUSLY.

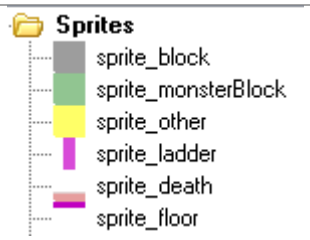
3 BACKGROUND AND ROOMS

Load the image **background_sky.png** as a **Background**. Make a new room called **room1** and set it to use that background. Make sure it is Tiled horizontally. Duplicate it to make **roomEnd**	
Load the image **blocks1.png** as a background. Click **Tile Set** checkbox ***TILES ARE SMALL BACKGROUND OR FOREGROUND IMAGES DESIGNED TO BE "PAINTED" ON A ROOM.*** Set the tile properties to the values shown to the left. The goal is to get the black tile borders to perfectly line up with the square tiles: 	 
Also, load the image **blocks2.png** as a background. Click **Tile Set** checkbox and use the same **Tile Properties** settings as you did for blocks1	

4 SPRITES

Load the sprites for the various “terrain” objects in the games.

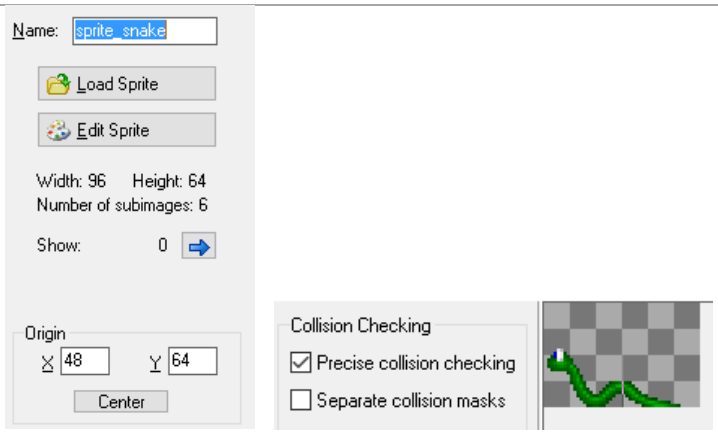
Each sprite should have its **Origin** at **0, 0** and should have **Precise Collisions** (Actually, the sprites that take up the full square do not need precise checking, but the rest definitely do.)



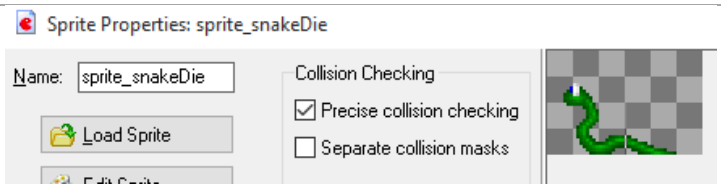
Load **snake.png** image as **sprite_snake**. Then add snake_1.png to snake_5.png as extra subimages.

Set **precise collisions** and place the **Origin** one pixel BELOW the center of the sprite (48, 64).

AN ORIGIN BELOW THE SPRITE TENDS TO WORK WELL FOR GAMES THAT INVOLVES “STANDING” ON GROUND. WHEN AN OBJECT IS PLACED ON THE GROUND IT IS JUST BARELY TOUCHING AND NOT INSIDE THE GROUND.

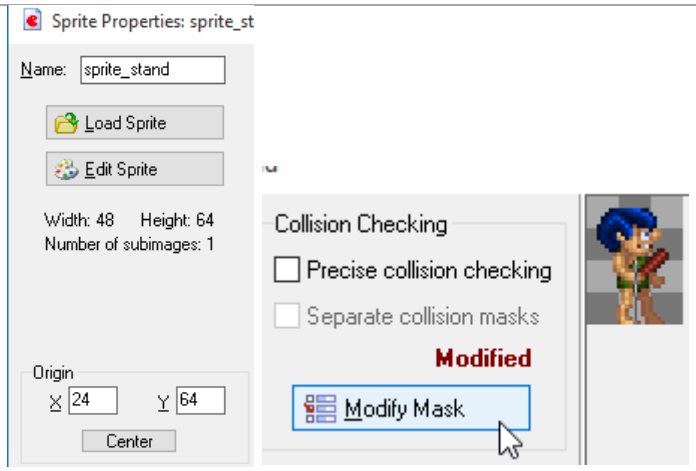


Do the same to make **sprite_snakeDie** from the 6 images of the snake disappearing.



Bring in **stand.png** as a sprite.

Set the **Origin** to one pixel below the middle of the character (24, 64)

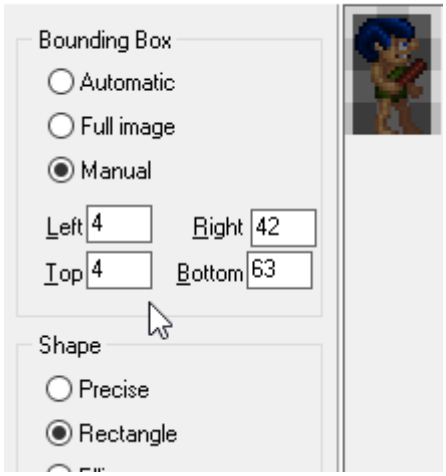


Click **Modify Mask**. Click the **Rectangle** radio button and type in the coordinates shown to the right.

This defines the gray box as the part of this sprite that collides.

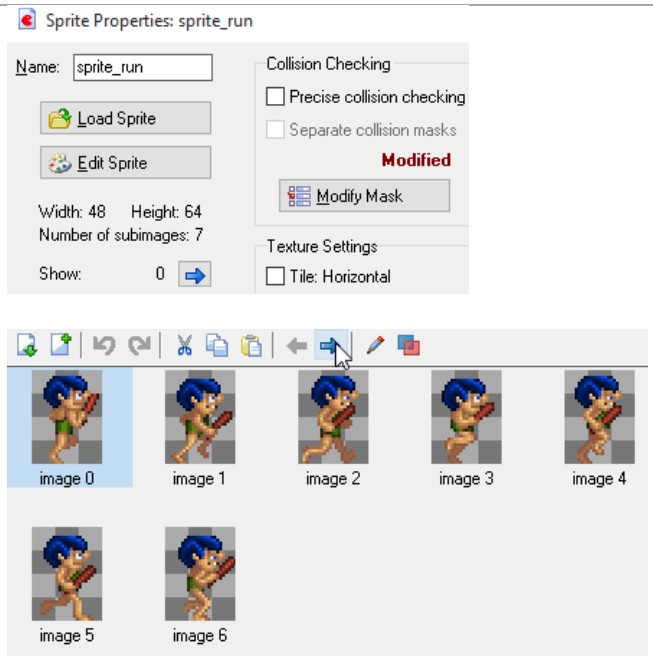
ANIMATED SPRITES THAT CHANGE A LOT FROM FRAME TO FRAME CAN CAUSE NIGHTMARES WHEN USED WITH PRECISE COLLISIONS. ONE SECOND YOU ARE NEXT TO A WALL AND THEN YOU TURN AROUND AND PART OF YOUR HAIR IS TOUCHING THE WALL AND NOW YOU ARE STUCK INSIDE IT. DEFINING A RECTANGULAR OR ELLIPTICAL COLLISION AREA CAN USUALLY ELIMINATE THIS ISSUE.

and

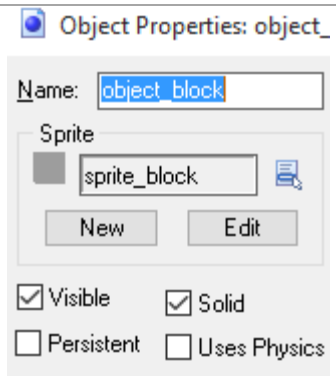


Bring in **jump.png** as a sprite and do the same steps for **Origin** and the **collision mask**.

We want every “player” sprite to use the same collision mask so that the process of switching a sprite can never cause a new collision to happen.

Bring in **run.png** as a sprite. Then add **run_1.png** to **run_6.png** as subimages. Set up the same **Origin** and **collision mask**. Make sure the images are in an order that looks somewhat natural when you preview it. You can use the blue arrow above the subimages to change the order if need be.	
Do the same for in **climb.png** to **climb_2.png** and **die.png** to **die_11.png** Make sure to use the same **Origin** and **collision mask** as for the other player sprites	

5 BASIC MOVEMENT

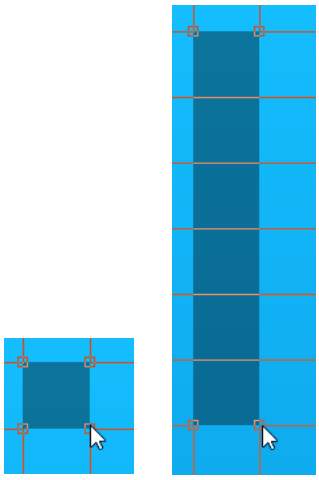
Make an **object_block** that is **Solid**. For now it will be visible, later we will turn that off. ***WHEN TWO THINGS COLLIDE AND ONE OF THEM IS SOLID, THEY ARE BOTH RESET TO THEIR POSITION JUST BEFORE THE COLLIDED.***	

Go to **room1**

Make sure that the Snap to grid is enabled and that the **snapX** and **snapY** are both **32**. This will make it much easier to design the room.

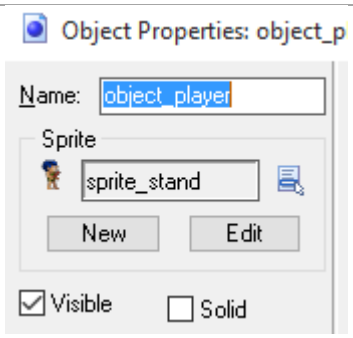
Place an object_block

Grab a corner and stretch it out. This is much easier than drawing hundreds of individual blocks.

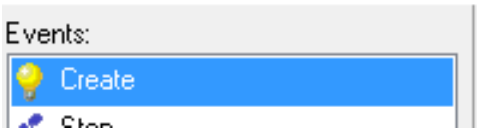


Stretch one block to be the left side, one to be the right, one to be the top and two to make a bottom with a hole (hole should be 5 or fewer squares wide). Place another block as an obstacle to jump over:

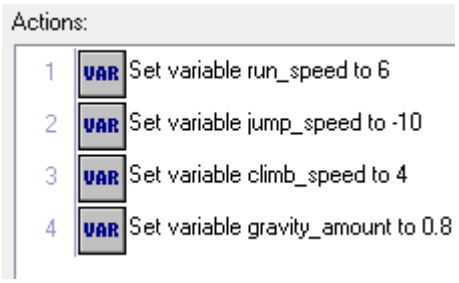
Make an **object_player**



The **Create** event should set variables that will control the speed of the player. Making them variables makes it easier to adjust them in one place. It also makes it much easier to make a powerup that makes the player move faster or jump higher.



We will use **run_speed**, **jump_speed**, **climb_speed**, and **gravity_amount** to control the player's movement actions.

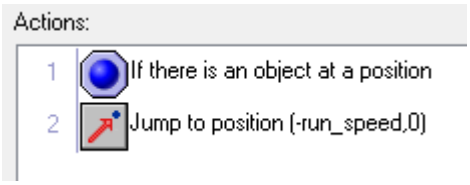


ANYTIME YOU TYPE THESE VARIABLE NAMES WATCH OUT FOR THE SPELLING!

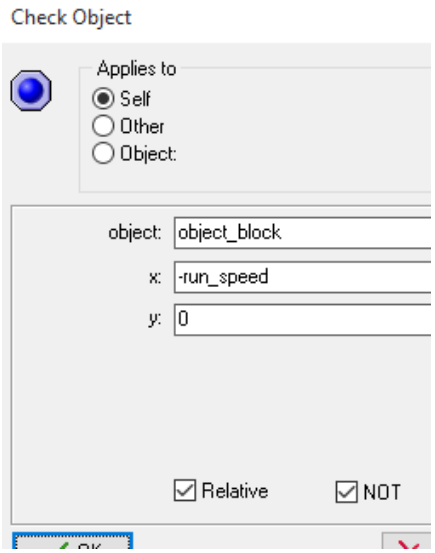
The **Left** keyboard event :

If at **relative position** (**-run_speed**, **0**) there is **not** object **object_block**

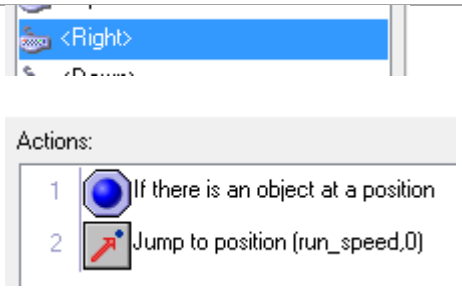
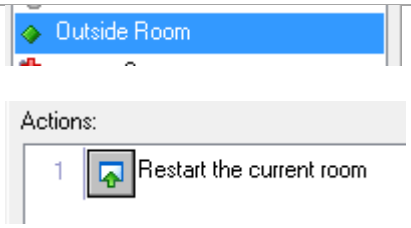
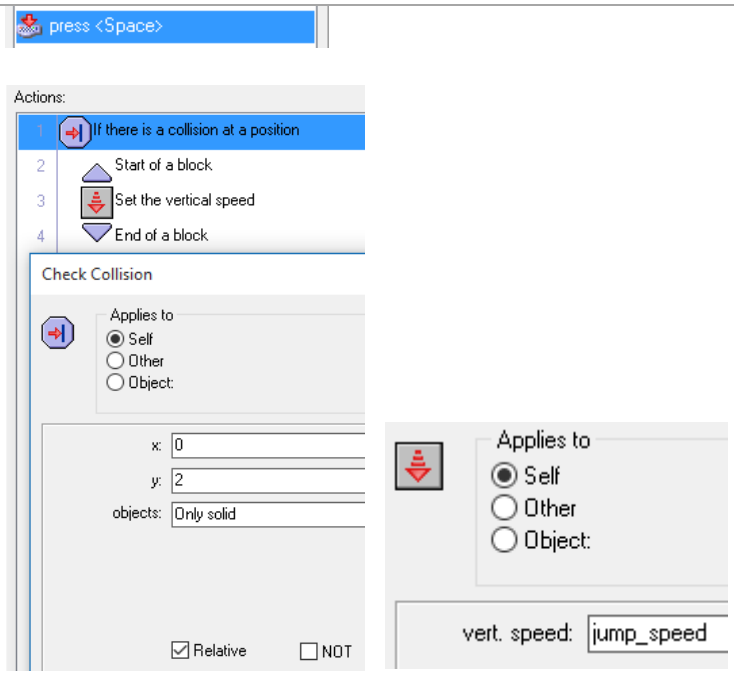
- Jump **relative** to position (**-run_speed, 0**)



The idea is we check to see if there is a block "run_speed" pixels to the left (-x is left, +x is right). If not, we jump there. The keyboard event does this every step so it looks like we are moving constantly.



NOTE: THIS STYLE PLATFORM MOVEMENT IS THE EASIEST TO PULL OFF. IT IS POSSIBLE TO MAKE MORE COMPLEX MOVEMENT WHERE YOU ACCELERATE UP TO MAX SPEED AND TAKE A SECOND OR TWO TO STOP MOVING WHEN YOU STOP PRESSING A BUTTON (MARIO STYLE). BUT THAT STYLE MOVEMENT REQUIRES A GOOD BIT MORE COMPLEXITY AND CODE.

Do the same for keyboard **Right**, but use **run_speed** in the **Check Object** and **Jump to Position** (+ means right, - means left)	
Place a player in the room and test the movement. At this point you cannot fall or jump, but you should move side to side and not get stuck in walls.	
Make leaving the room restart the current room.	
KeyPress Space should check to see if there is something solid just below the player (2 pixels below makes things a little more forgiving than just 1 pixel). *We only want the player to be able to jump if they are on the ground.* If so, set the **vertical speed** to **jump_speed**	

Without gravity, the player will just fly up. So we also need to turn that on. But gravity should only be on when the player is in the air… we cannot just turn it on when it is created.

GRAVITY FOR THINGS STANDING ON THE GROUND SHOULD NOT BE ON. IT WILL CAUSE THEM TO KEEP TRYING TO ACCELERATE DOWN AND COLLIDING WITH THE FLOOR. THIS WORKS WELL IN REAL LIFE, BUT NOT IN GAMEMAKER.

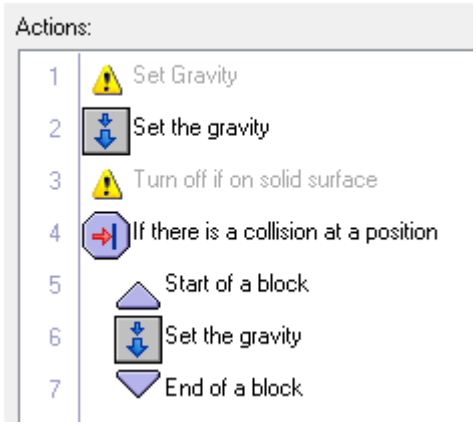
Instead, in **Step** we will check to see if gravity should be on because the player jumped or walked off a cliff.

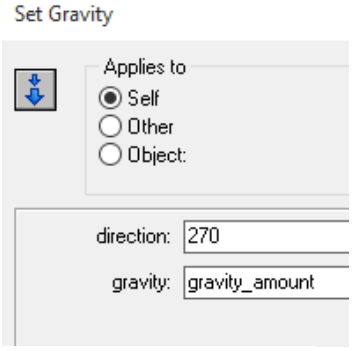
We will assume gravity needs to be on and then turn it back off if we realize there is something solid below the player.

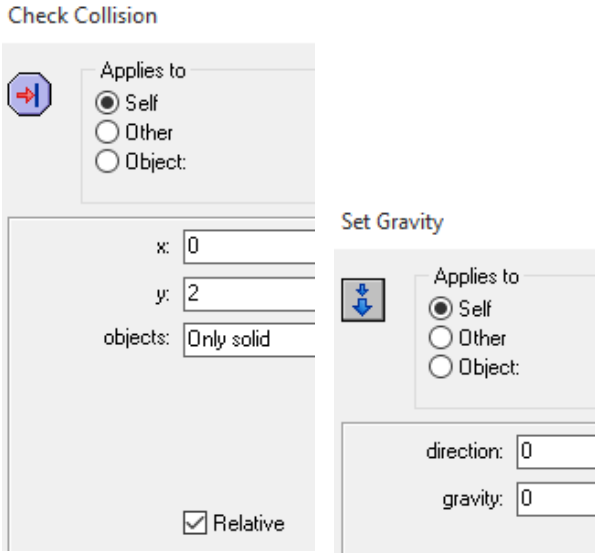
Step Event:
Set the gravity to **gravity_amount** in direction **270**
Gravity amount is never going to be relative (it is either on or off, the amount does not increase or decrease)
(Assume it needs to be on)

If **relative** position (**0, 2**) gives a collision with **Only solid** objects

- **Set the gravity** to **0** in direction **0**
  (Ooops, standing on something, turn it off)







Last we need to stop the player when they hit a block. In **Collision** with **object_block**

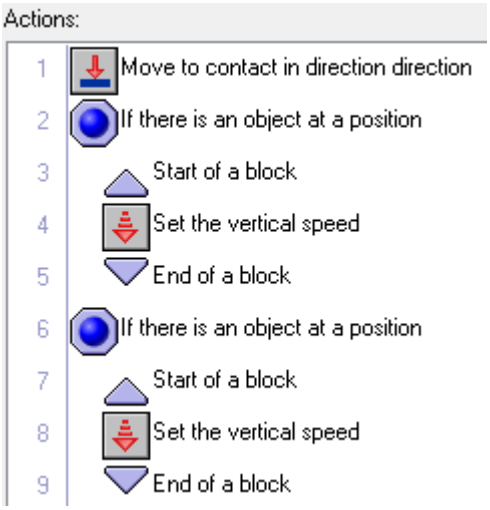
First **Move to Contact** in direction **direction** maximum distance **speed**

When you collide with a solid object you bounce back to where you used to be... this ooches us right up next to the thing we hit.
*We are using the built in variables direction and speed to say "keep moving your current **direction** until you are just touching the block; don't go more than your **speed** number of pixels"*

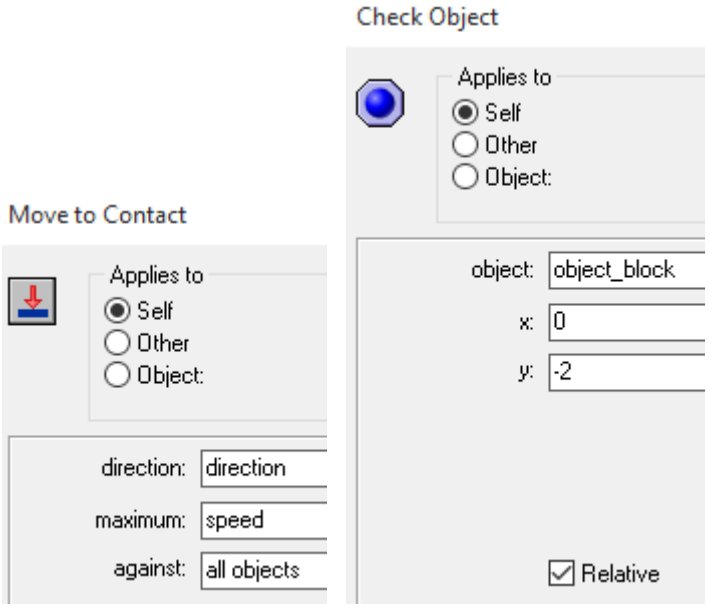
Now we need to turn off the vertical speed – but only if there is a block right above or below you (not if you slammed sideways into a wall).

- if at **relative** position (**0, -2**) there is object **object_block**
    - **set the vertical speed** to **0**
- if at **relative** position (**0, 2**) there is object **object_block**
    - **set the vertical speed** to **0**

BASIC JUMP RECIPE:
WHEN STANDING, GRAVITY IS OFF AND YOU CAN JUMP. WHEN YOU JUMP, START MOVING UP AND TURN ON GRAVITY. WHEN YOU HIT THE FLOOR, STOP VERTICAL MOTION AND KILL GRAVITY.



Details:

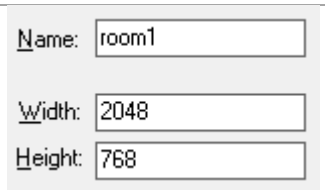


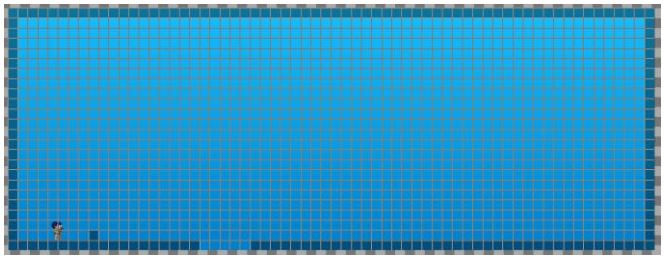
Try the game again. You should be able to jump over the obstacle and the pit. Make sure you can move normally after landing and you fall if you walk off an edge.

6 VIEWS

Reopen **room1** and make it **2048** wide.

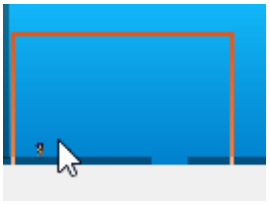
Reposition the walls/floors so it looks like the picture shown to the right.





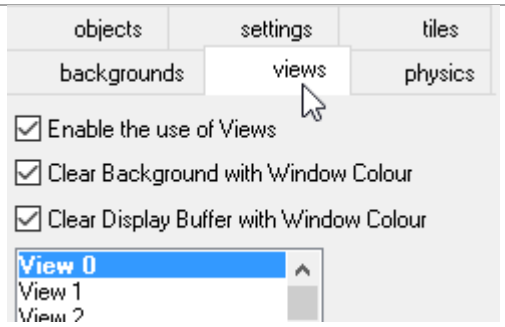
HINTS:

- ***YOU CAN USE THE MAGNIFYING GLASS ICONS ABOUT THE ROOM WINDOW TO ZOOM IN AND OUT.***
- ***YOU CAN MOVE THE ROOM AROUND BY HOLDING SPACE WHILE CLICK DRAGGING.***
- ***YOU ALSO CAN DRAG AROUND THE WINDOW SHOWN IN THE MIMIMAP TO SCROLL AROUND THE ROOM:***



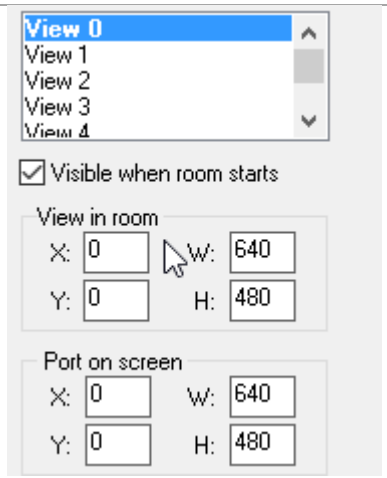
Click the **Views** tab and **Enable the Use of Views**

A VIEW IS A PICTURE OF ALL OR PART OF A ROOM THAT IS TAKEN EACH STEP AND DISPLAYED TO THE PLAYER. IT IS HOW WE CAN SHOW ONLY PART OF A ROOM TO A PLAYER OR SHOW A MINIMAP VIEW OF THE ROOM.

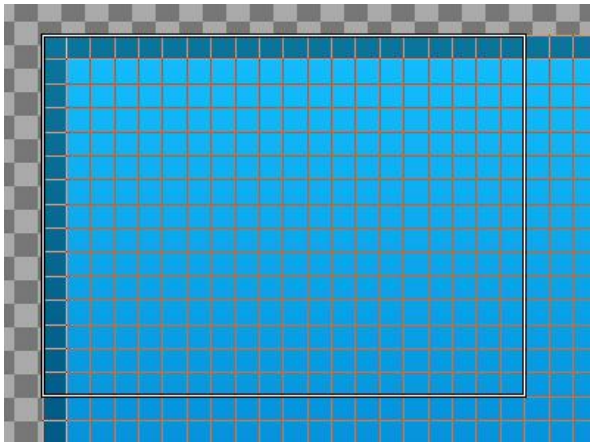


The initial settings should look like what is shown to the right.

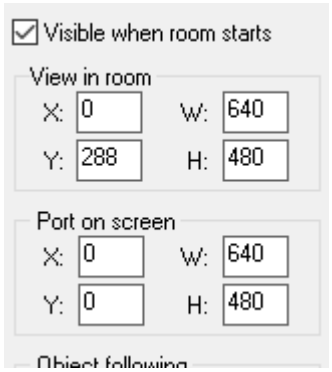
It says “take a picture that starts at 0, 0 and is 640x480” (The **View in room** settings).
Then “draw it on the screen 640x480 pixels wide at 0, 0” (The **Port on screen** settings)

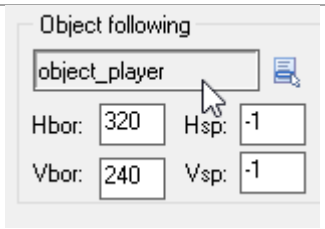


A white outline shows what the view covers:

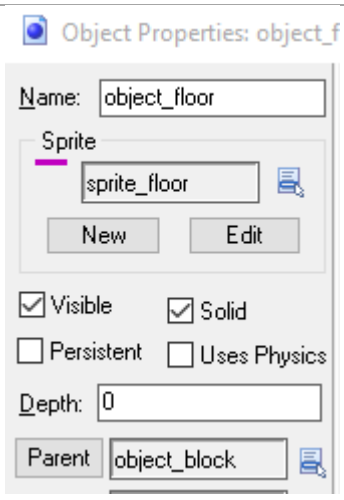


Change the **View in room** so that it includes the player



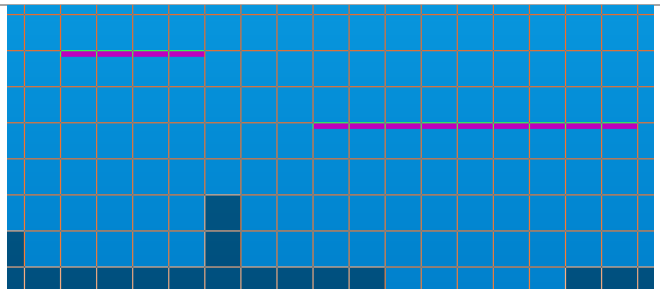
Try the game. Notice you have a smaller game window now. Try changing the **Port on Screen** to 1280x960 (assuming your monitor is that big at least). And run the game. Now you are taking a picture that is 640x480 and blowing it up to 1280x960 so everything looks blocky. Try changing the **Port on Screen** to 320x240. Now the picture gets shrunk in half when we draw it for the player. Reset the **Port on Screen** to 640x480. Generally you want the View size (picture size) to match the Port size (drawing size)	
Also, the view does not follow the player around. To fix that, go back to the Views tab of the room and select **Object following** to be the player. The **Hbor** and **Vbor** are how close the object can get to an edge before the view scrolls. Set those to half the view width and height (320x240) to keep the player in the center. The **Hsp** and **Vsp** can limit the speed at which the view can move (if you want to slowly pan across the room say). -1 means "no limit" to the speed.	
Try the game – make sure you can run around and the camera follows you.	

7 PLATFORMS, LADDERS & EXIT

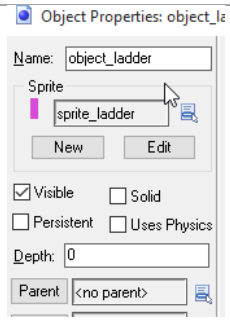
Make **object_floor** with **object_block** as its **parent** It should be **solid**. We don't need to add any extra code – player collides with block; floor "is a kind of block" because block is its parent; thus player collides with floor. The floor just has a smaller sprite.	

Place a few object_floor's in as platforms. Make sure you can jump on them and move around. Hitting the platforms from below or the side will stop you.

MAKING PASS THROUGH PLATFORMS YOU CAN JUMP UP THROUGH AND THEN LAND ON IS POSSIBLE, BUT REQUIRES SIGNIFICANTLY TRICKIER LOGIC.

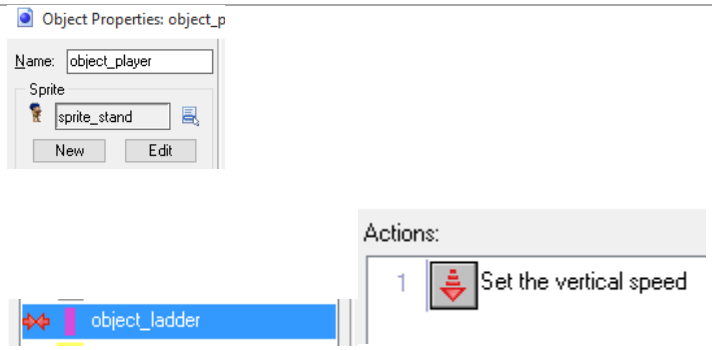


Make **object_ladder** it is not solid – the player can climb through it



Now go back to object_player.

Colliding with a ladder should set the **vertical speed** to **0**
This will stop any jump/fall that is happening.



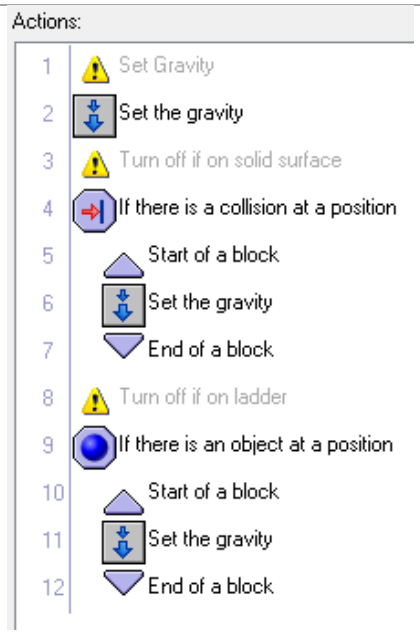
We also need to turn off gravity while hanging on a ladder.

Go back to the **step event**. Add a check to see if there is a ladder right where the player is and if so, turn off gravity.

If at **relative** position (**0,0**) there is object **object_ladder**

- **Set the gravity** to **0** in direction **0**

The overall logic is:
Assume gravity is on.
If there is floor beneath us, turn it off.
If we are on top of a ladder, turn it off.

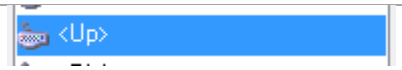


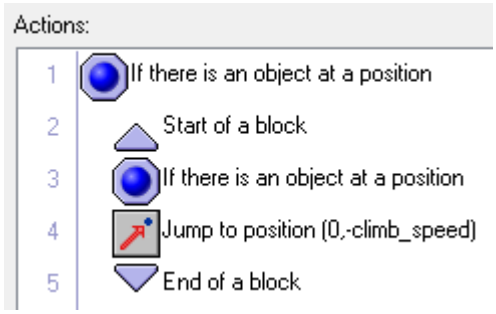
If you are on a ladder and there is not a block right above you, you should be able to climb up.

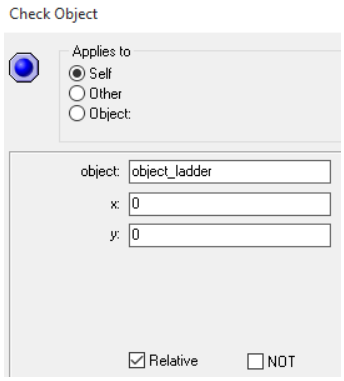
Up button should do:

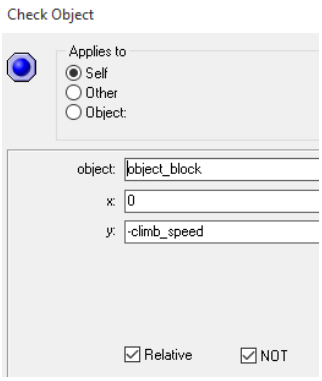
- If at **relative** position (**0,0**) there is object **object_ladder**
  - If at **relative** position (**0, -climb_speed**) there is **not** object **object_block**
  - **Jump relative** to **position** (**0**, **-climb_speed**)

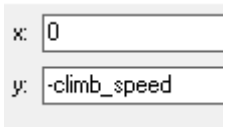
-y is up, +y is down





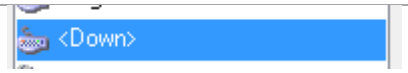


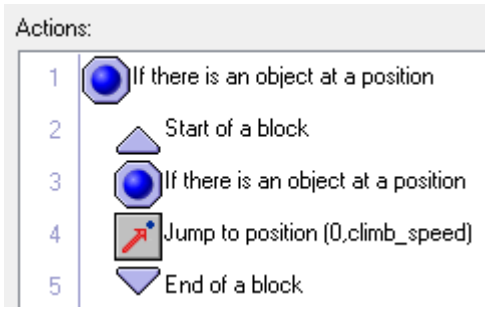




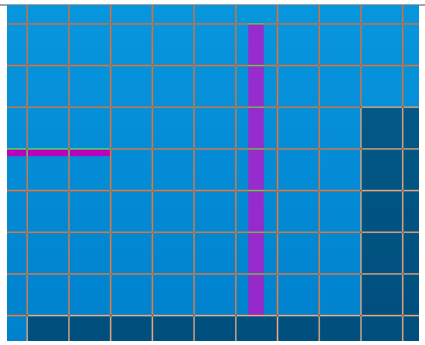
Down should be the same, but with **climb_speed** instead of **-climb_speed**
(+: down, -: up)

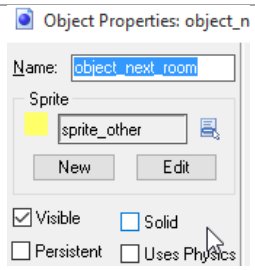
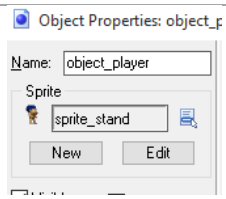
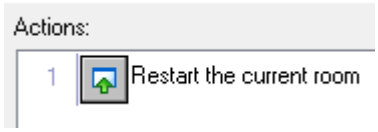
- If at **relative** position (**0,0**) there is object **object_ladder**
  - If at **relative** position (**0, climb_speed**) there is not object **object_block**
  - **Jump relative** to **position** (**0**, **climb_speed**)



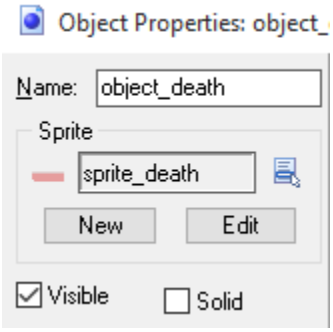
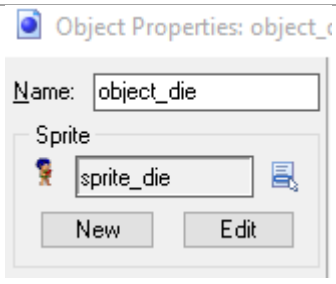


Place a ladder in the room. Try jumping on to it and climbing up and down.



Make an **object_next_room**	
Go back to **object_player** When it **collides** with **next_room**, send the player to the next room.	  
Place an **object_room_next** at the far side of the level. (Your **roomEnd** is proabably empty… that is OK – you can make more rooms or a “Game Over – You Win” message later)	

8 OBSTACLES

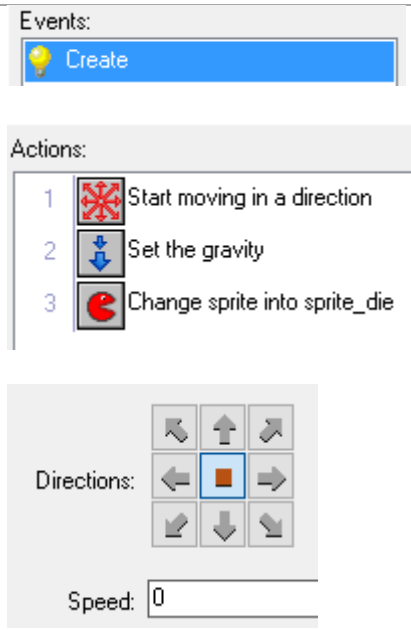
Make **object_death** We will use it to make “spike pits” or other areas that can kill the player if they fall into.	
Make **object_die** It will show the player’s death animation	

In **Create**:

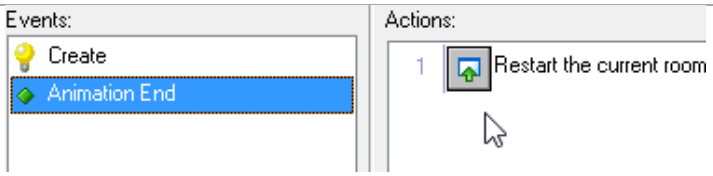
Start moving in no directions with **speed** set to **0**
Set Gravity to **0** with direction **0**
Set the sprite to **sprite_die** with subimage **0** and speed **0.5**

The player will be turning into this object – we need this code to make sure that they are stopped once they turn into the dying object.

Playing the sprite frames at 0.5 speed makes the animation last a little longer.

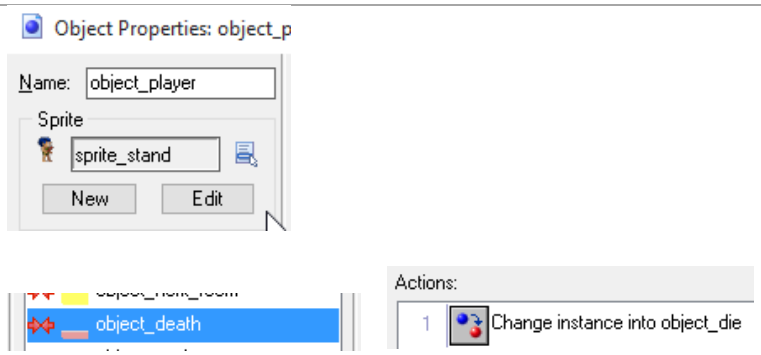


Animation End should restart the room

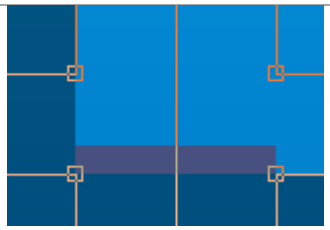


Go back to the player

Make **colliding** with **object_death Change Instance** to turn the player into an **object_die**
Set **Perform events** to **yes** to make sure we do the create code on the object_die



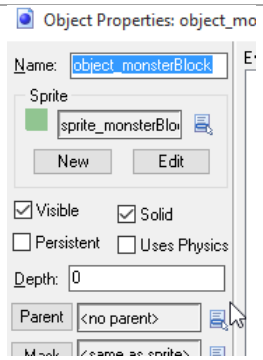
Go place an object_death in an easy place to test. Try colliding with it and make sure the player dies. Once you know it works you can move it to a place where the player might fall.



Make **object_monsterBlock**

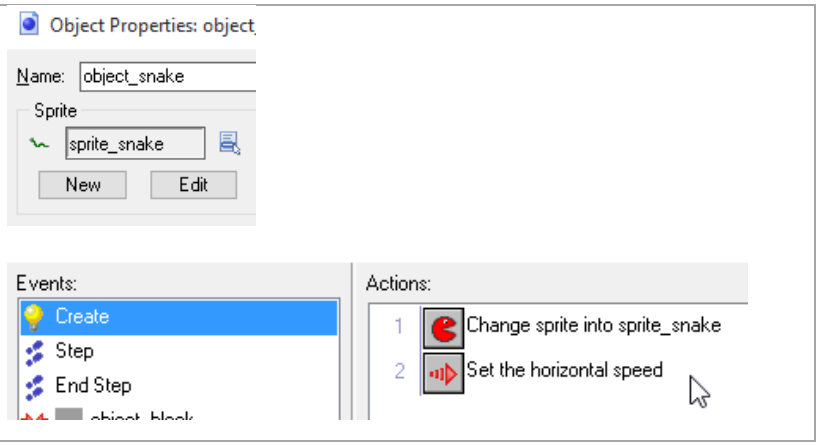
We will use it to make boundaries that monsters bounce off of but that do not affect players.

It should be solid, but not use the normal block as its parent (do not want the player to interact with it).



Make an **object_snake**

Create event:
Set the sprite to **sprite_snake** with subimage **0** and speed **0.5**
Set the **horizontal speed** to **3**

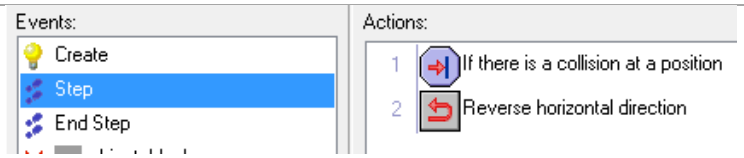


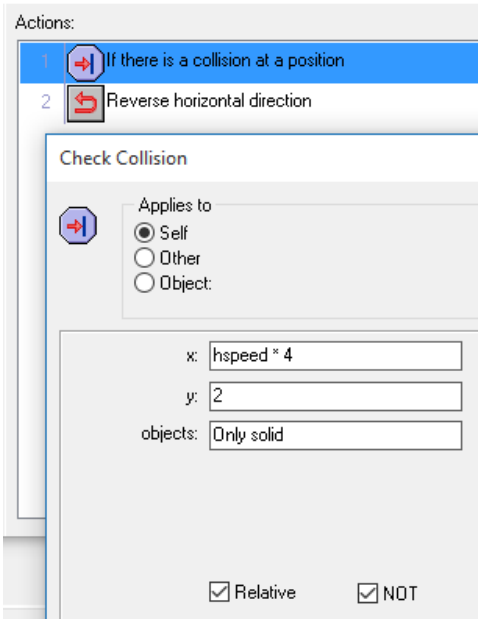
If the snake gets to an edge it needs to turn around. We will check for a collision 2 pixels down and about 12 forward from the current location (relative). If there is nothing there (Not a collision), that means the floor is gone and the snake needs to turn around.

Step event should check:
If **NOT** Check Collision x: **hspeed * 4** and y: **2 relative**
Reverse Horizontal Direction

For x: we are using **hspeed * 4** instead of 12 because hspeed will include the sign – when the snake is going right, its hspeed is 3. Once it turns and goes left, its hspeed is -3. By multiplying that by 4, we get either 12 or -12 for x and measure 12 pixels in front of whatever direction the snake is pointing.

hspeed *is the built in variable that has just the horizontal part of an object's velocity*
vspeed *has just the vertical part*





In **End Step** we will transform the sprite to go the right way. The sprite is facing to the left, so when moving right (hspeed > 0) we need to flip the sprite.

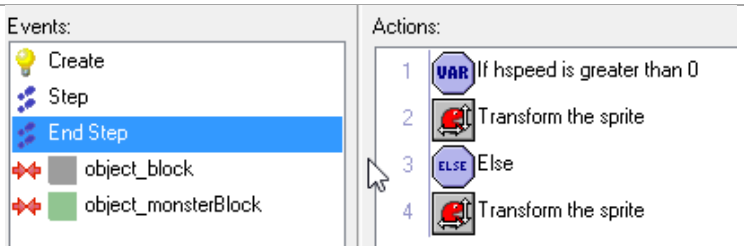
If **hspeed** is **greater than 0**

- **Scale the sprite** with 1 in the xdir, 1 in the ydir, rotate over 0, and **mirror horizontally** *(Show it flipped)*

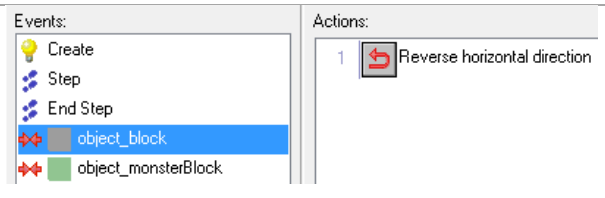
else

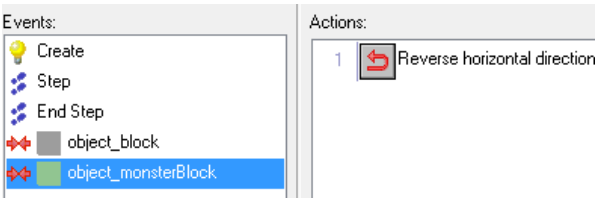
- **Scale the sprite** with 1 in the xdir, 1 in the ydir, rotate over 0, and **no mirroring** *(Show it normal)*

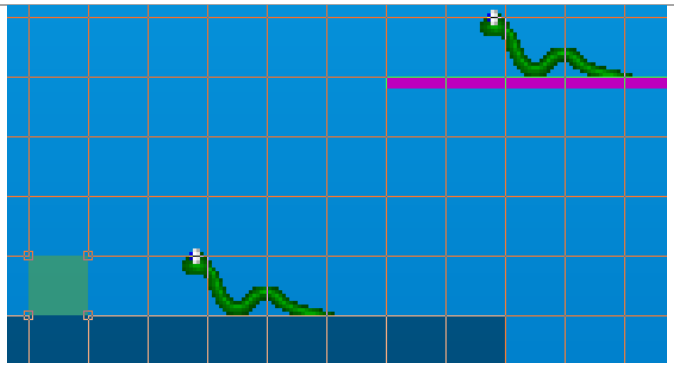
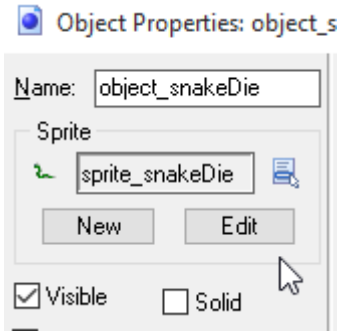
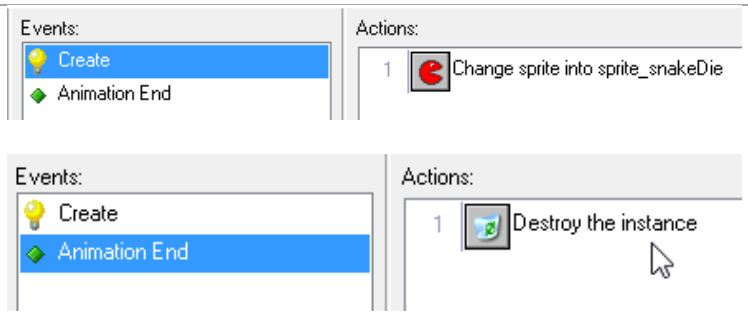
END STEP IS DONE AFTER MOST OTHER EVENTS, RIGHT BEFORE THINGS ARE DRAWN.



Finally, for both **object_block** and **object_monsterBlock**, **reverse horizontal** when we collide





Place some snakes in the room. Also place a monsterBlock or two. Make sure snakes go back and forth, bump off walls and turn around at edges. You may want to turn off views or make the view really large so you don't have to run around to see everything.	
Make an **object_snakeDie**	
When it is created, set its sprite to the same sprite but using speed **0.5** When the **animation ends**, **destroy it**	

Go back to **object_player**

When a player and snake collide, we will check their relative height on the screen (y position). If the player is at least 10 pixels above the snake, the snake is smashed and the player bounces up. Otherwise the player dies.

Note that comparing their y locations compares the Origins of their sprites. Since both have sprites with Origins at their "feet" this works well. If both had Origins at their head, we would have to compensate for the different heights of the sprites.

Collision Event with object **object_snake**:

If **y + 10** is less than **other.y**

- For **other** object (snake): **change the instance** into object **object_snakeDie**, **yes** performing events
- Set the **vertical speed** (for self, the player) to **jump_speed**

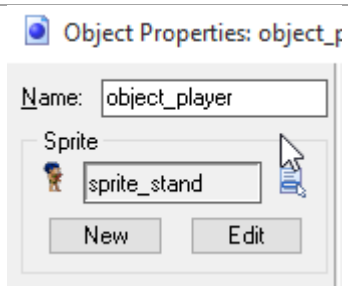
Else

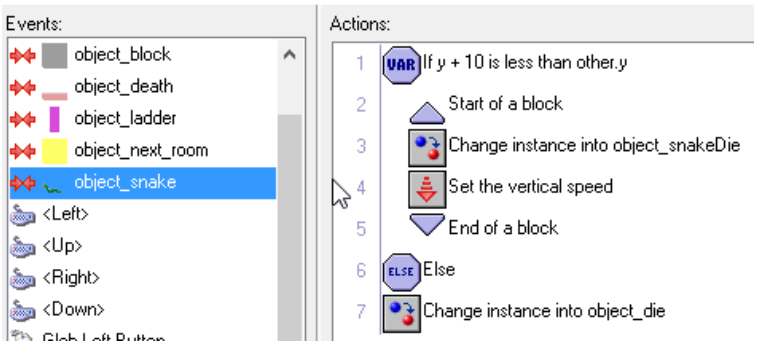
- **Change the instance** (self, the player) into **object_die**, **yes** performing events

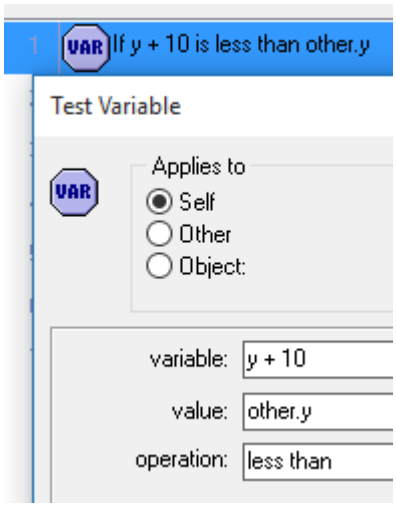
y gets smaller as you go up. So if one object has a smaller y than another, it is above the other.

"y" is player's y position. "y + 10" is 10 down from the player's y.

"other" here is the snake the player is colliding with. So "other.y" is the snake's y position.







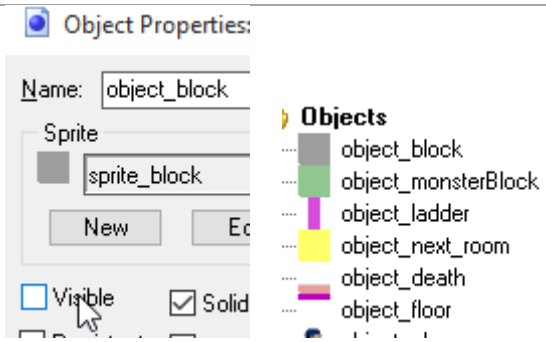
Go test that running into a snake restarts the room but landing on one kills the snake and bounces you.

9 TILES

We will use tiles to provide some visual appeal to the plain objects.

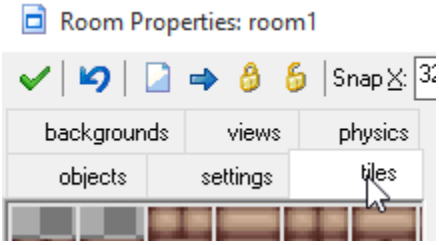
First go turn off **Visible** for the **object_block**, **object_death**, **object_ladder**, **object_floor**, **object_next_room** and **object_monsterBlock**

INVISIBLE OBJECTS SHOW IN THE ROOM EDITOR BUT NOT THE GAME.



Go to **room1**

Click the **Tiles** tab



Drag the separator bar between the tiles and the room preview to the left. You may want to make the whole window larger as well so you can see all the tiles and still see a good chunk of the room.

Click on a tile to use it.

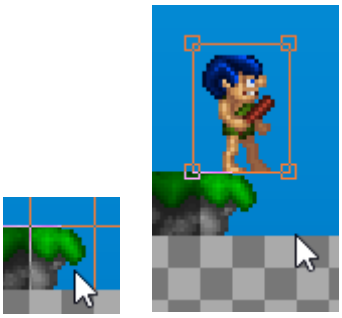
Then click in the room to place it. Tiles never interact with objects. They are just pretty window dressing to cover up simple objects.

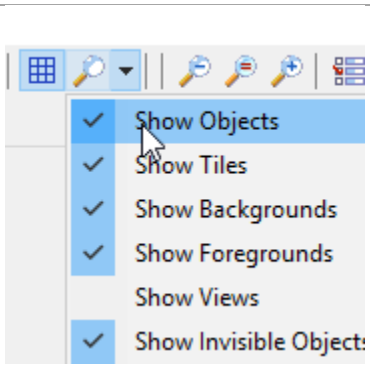
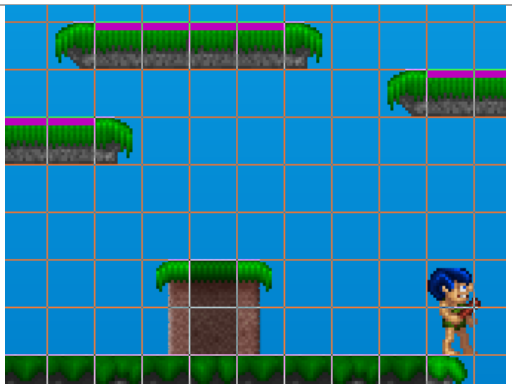
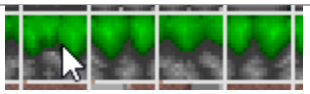
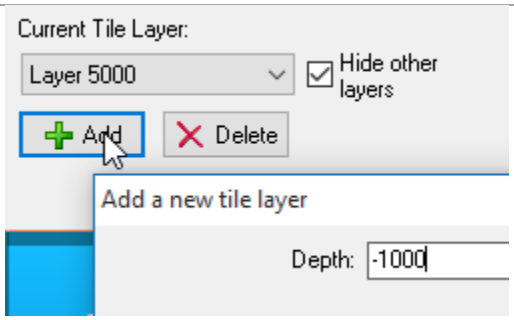
SHIFT-CLICK AND DRAG TO RAPIDLY "PAINT" A WHOLE ROW OF TILES.

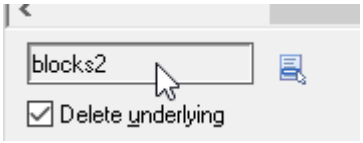
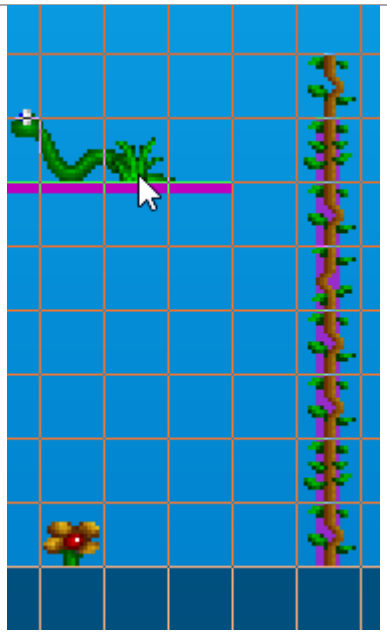
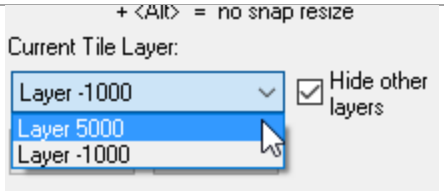
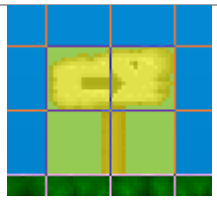
CNTRL-RIGHT CLICK TO RAPIDLY DELETE TILES

Click a different tile to switch what you are "painting" on the room.

Note that edges should have tiles one square past where the actual object_block is. Notice how the player extends almost a full square in either direction from its center. Right now, the player would still think that there is a collision with block below it (its collision area is almost the entire rectangle). A few more pixels to the right, and the player rectangle no longer would be over the block.



You can turn the visibility of objects or tiles on and off using this control. Try it out. Our objects are translucent, so you should be able to leave them on. But if they were opaque, you might need to turn them off to check your tiles.	
Try painting tiles over your floors, something like shown to the right:	
Try to not have large runs of the same tile – many of the tiles have multiple similar options. Do a pass where you cover the floor with one style, then pick something similar and replace a few with that. Then pick a third style and replace other squares with that one…	
You can add multiple tile layers. ***HIGHER DEPTH LAYERS ARE DRAWN FIRST, THEN LOWER NUMBER LEVELS ARE DRAWN ON TOP OF THEM.*** ***THIS IS THE SAME DEPTH SCALE USED BY OBJECTS. NEGATIVE DEPTH WILL DRAW IN FRONT OF MOST OBJECTS.*** Add a layer at -1000.	

Use this negative depth layer to paint things that should be on top of the player and snakes. Tufts of grass, flowers and the ladder tiles (especially the ladder) all look nice in front of objects. Many of these images are from the second tileset:  Note that the ladder tiles extend a square past the actual ladder. This will prevent the player from standing on the ladder object with just two pixels of their toes covered by the ladder tile.	
Use the drop down to switch between layers and the checkbox to hide layers other than the one you are working on or reveal them all.	
There is no magic way to draw multi tile patterns. You just have to place the tiles one by one for things like signs, clouds, etc… *If you were making your own graphics, you could make multiple tile sheets that each used different size tiles.*	 

10PLAYER SPRITE

THIS STEP IS OPTIONAL: We have saved changing the player sprite for last because it is fairly complex. Also, it just isn't essential for the gameplay. Yes, it will look slicker if the sprite is animated, but that is much less important than everything we have done up until now.

Deciding exactly which sprite to draw depends on whether we are in the air, whether we are trying to go left or right, whether we are on a ladder, etc… The logic is best handled all in one place.

An **End Step** event is a good spot to select the sprite after all the movement code has run.

You do not need to put the comment blocks (yellow triangles) in – they are just to help you see what is going on. Though any time you do something this complex, leaving reminders of what you were doing is a good idea.

Part 1: Turn on or off sprite mirroring depending on if pressing a direction:

If expression **keyboard_check(vk_left)** is true

- **Transform Sprite** with scale 1 in the xdir, 1 in the ydir, rotate over 0, and **mirror horizontally**

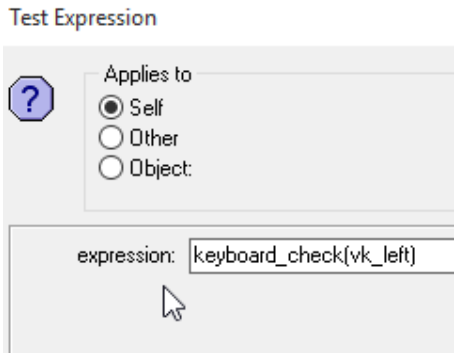
if expression **keyboard_check(vk_right)** is true

- **Transform Sprite** with scale the sprite with 1 in the xdir, 1 in the ydir, rotate over 0, and **no mirroring**

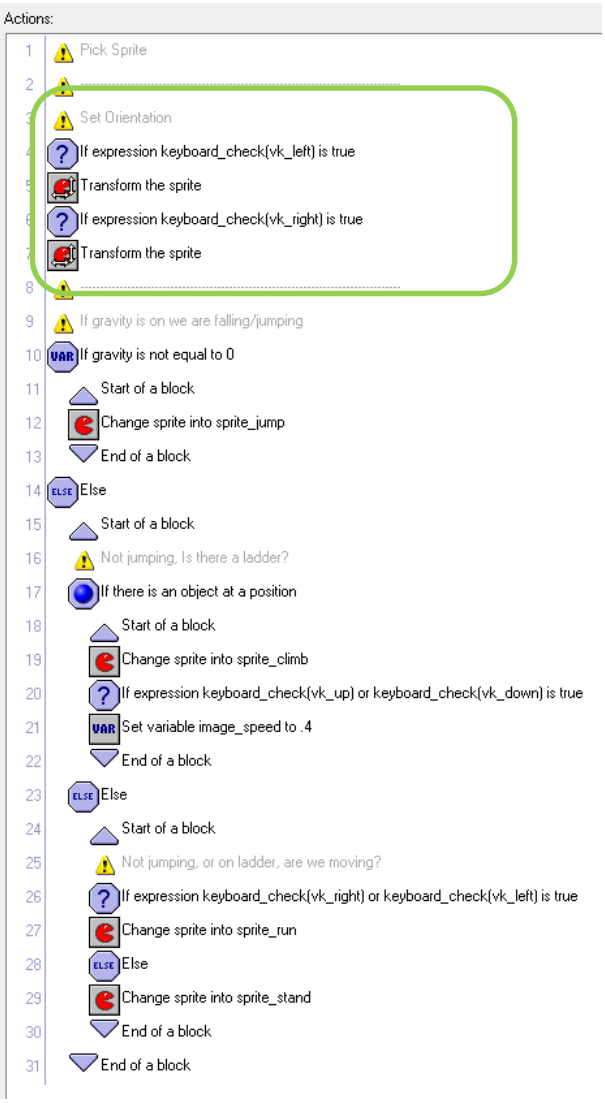
If neither is pressed, we will just keep using whatever was last set

keyboard_check(BUTTON) is true if that button is being pressed. You need to use vk_left to mean “left arrow” and “vk_right” to mean right arrow.

This should look like:



[See here for more info on keyboard_check]



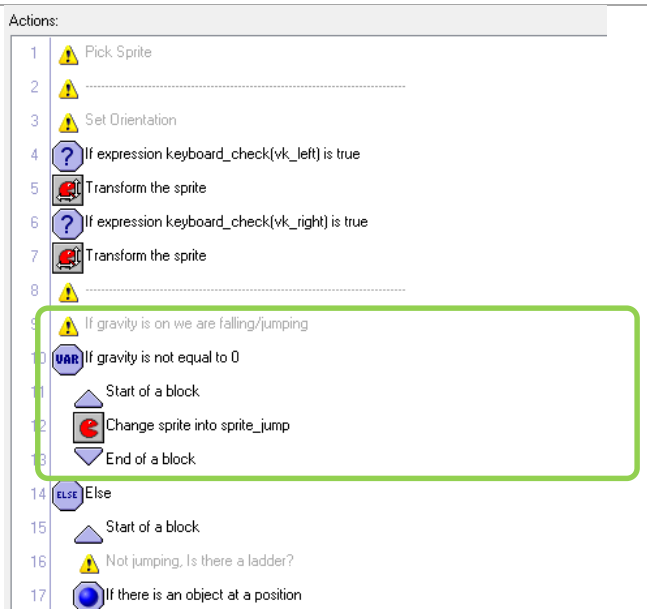
Part 2: If they are in the air, show jumping sprite. We know they are in the air (falling/jumping) if gravity is on.

If **gravity** is **not equal** to **0**

- **Set the sprite** to **sprite_jump** with subimage **-1** and speed **1**

SETTING SUBIMAGE TO -1 SAYS "DO NOT CHANGE THE SUBIMAGE". IF YOU SET THE SUB IMAGE TO 0 DURING ANIMATIONS, IT WILL KEEP RESETTING THE ANIMATION BACK TO THE FIRST FRAME AND YOU WILL NEVER SEE THE ANIMATION PLAY.

IF YOU ARE SETTING AN ANIMATED SPRITE BUT MIGHT NOT BE ACTUALLY CHANGING WHAT SPRITE YOU ARE USING (WAS RUNNING LAST STEP, AM STILL RUNNING), YOU NEED TO USE -1 FOR SUBIMAGE SO YOU DON'T RESET TO FRAME 0 EVERY STEP.



...

Part 3: Not in air… are we on a ladder?

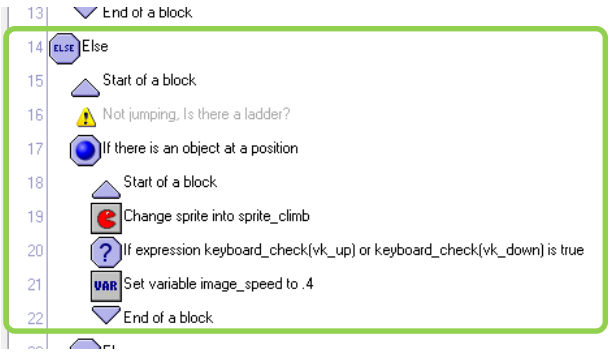
else

- If at **relative** position (**0,0**) there is object **object_ladder**
  - **Set the sprite** to **sprite_climb** with subimage **-1** and speed **0**
    This sets to climb sprite… now check to see if we need to animate it
  - If expression **keyboard_check(vk_up) or keyboard_check(vk_down)** is true
    - **Set variable image_speed** to **0.4**

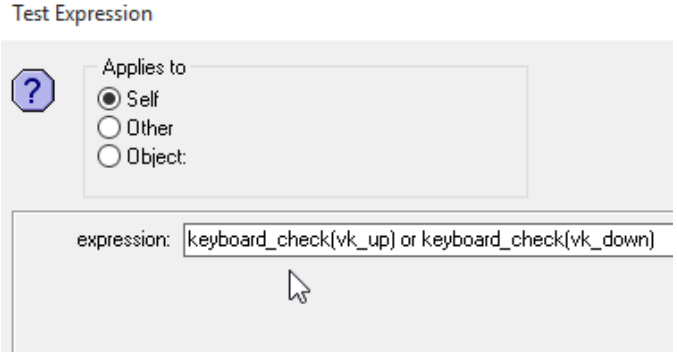
    This makes the animation play if we are pressing up or down.

IMAGE_SPEED IS THE BUILT IN VARIABLE FOR HOW FAST THE FRAMES OF A SPRITE ARE PLAYING. SETTING IT DIRECTLY IS A NICE WAY TO PLAY OR STOP AN ANIMATION WITHOUT HAVING TO WORRY ABOUT WHICH SPRITE EXACTLY YOU ARE WORKING WITH.

...



...



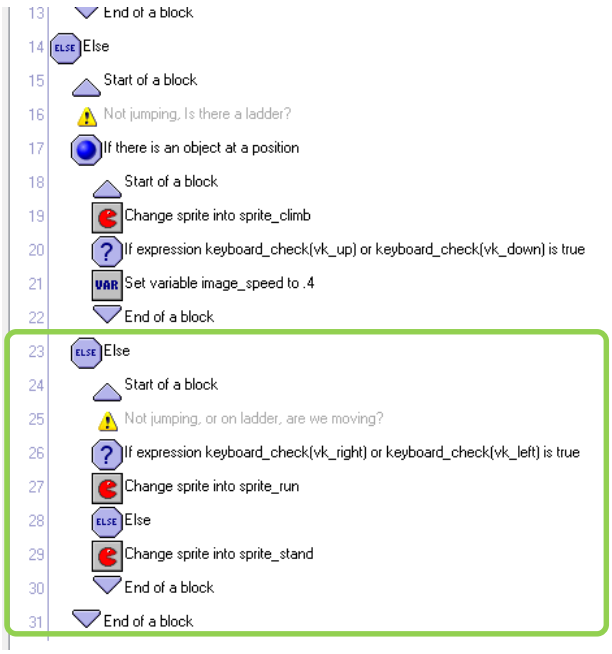
Part 4: Not in air and not on ladder, must be standing or running

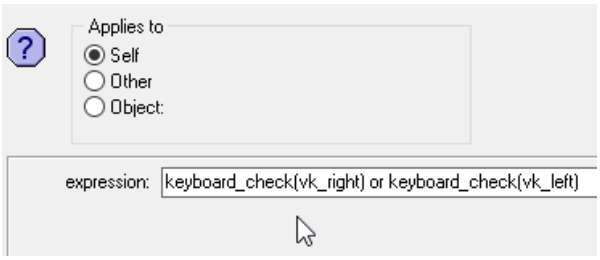
- **else**
  - **If** expression **keyboard_check(vk_right) or keyboard_check(vk_left)** is true
    - **Set the sprite** to **sprite_run** with subimage **-1** and speed **0.5**

    If left or right is pressed, we are running… so play that
  - **else**
    - **Set the sprite** to **sprite_stand** with subimage **-1** and speed **0**

    Otherwise standing

…





Go back to the start of this section… make sure all the actions are laid out correctly.
Then go test out the animations. If something is wonky, double check the appropriate section of code.

If everything ends up horribly broken, use an Exit this Event action at the start to just turn all the code off so you can focus on other stuff.

…

11 Extras

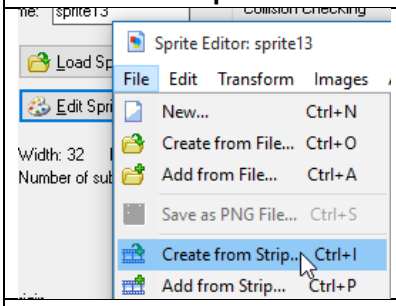
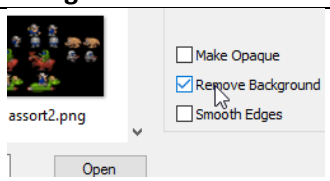
Time to improve on the game.

Before you start making changes, make a backup copy of your project. Copy the WHOLE Platformer.gmx folder, or whatever you called it, to another location. You should spend a couple of hours experimenting and trying to add some features to the game.

Focus on adding new features (objects and behaviors) not on endless tweaking of the tiles or building out a ton of rooms.

The ExtraArt folder has additional sprites you can use. If you are familiar with basic graphics programs you can also use the image editor in gamemaker to customize sprites. Do not spend lots of time trying to get graphics just right – I want you to focus more on making things happen than looking pretty.

Note that the sprites are arranged in sheets. To get one sprite out of a sheet:

Make a Sprite, then **Edit** it. In edit window, do **File → Create from Strip**	Pick the sheet. You probably want to **Remove Background** color
	

Set the **width** and **height** large enough to get the image you want. **Click with the mouse** to position the selection rectangle. Or use the vertical/horizontal offset to fine tune. Note that the selection border inverts colors it is over. These pixels ARE included. The image shows the watermellon selected with one empty pixel on each side and no empty space at the bottom of the sprite.

IDEAS:

You do not have to do all of these – you do not even have to do any of them if you have your own ideas. As you add features, try to think about what is going to add "fun" to the game. Don't worry about adding features like lives and score unless you think they are going to add "fun" to the game. Focus on changes to the gameplay.

- Pickups that change something about the player (speed, vulnerability)
- Experiment with different movement code
- Bad guys that appear/start moving when you get too close. Start with an invisible object that has a large circle for a sprite (200+ pixel wide circle). When it collides with the player, it turns into the bad guy and starts moving.
- Portals (touching obj_portal_red could make the player move to obj_portal_blue.x and obj_portal_blue.y)
- You can't jump on bad guys, but have an attack. It needs to be balanced so you can't just easily kill all the snakes to get through the level. Maybe something that creates a small, quick explosion in front of you (easier than a

fancy club swing) that kills snakes, but there is a cooldown so if you miss on the first try the snake will probably get you.