1 Amazed Guide

2 Setup

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

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

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 Room 1

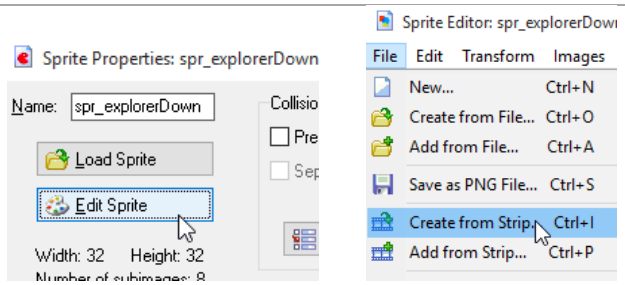
Load the image **back_sand.png** as a **Background**. Make a new room called **room1** (I used 640x480 for the size, you can make bigger or smaller if you like) and set it to use that background. Make sure it is **Tiled** horizontally and vertically.	

4 Sprites

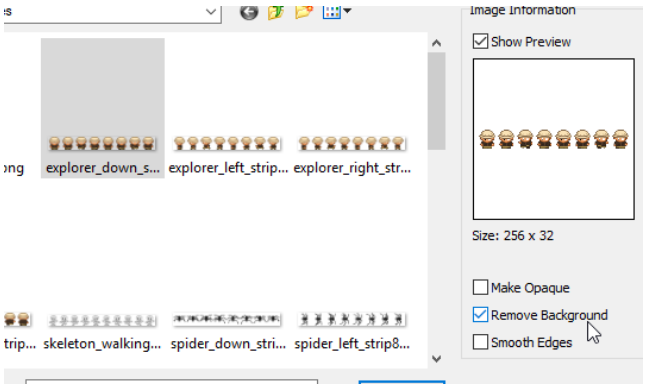
Load the sprites for the objects shown to the right. Each sprite except spr_dark should have its **Origin** at **0, 0** and should **NOT** have **Precise Collisions.** (We want everything to "take up" a full 32x32 square). **spr_dark** should have its **Origin** centered	

Create a new sprite **spr_explorerDown**

Edit the **Sprite**, then chose **File→Create From Strip**



Fine the **explorer_down_sprite**. You don't have to remove the background as this image has a transparent background. But if there was a solid color behind the explorer you can use Remove Background to delete it automatically.

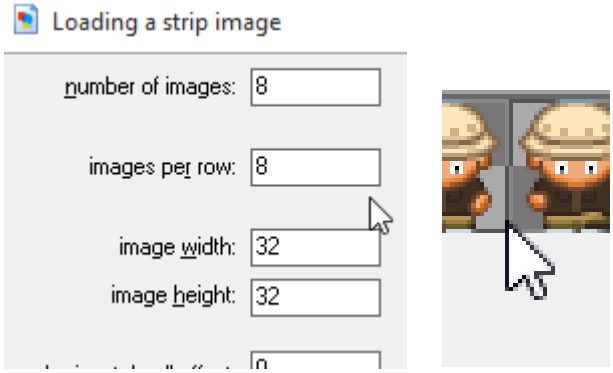


Then you will be asked how to chop up the strip. We have **8** images, and they are all in one row. Each image is **32x32** and there is no space between them.

Set the options as shown to the right (all other numbers are 0).

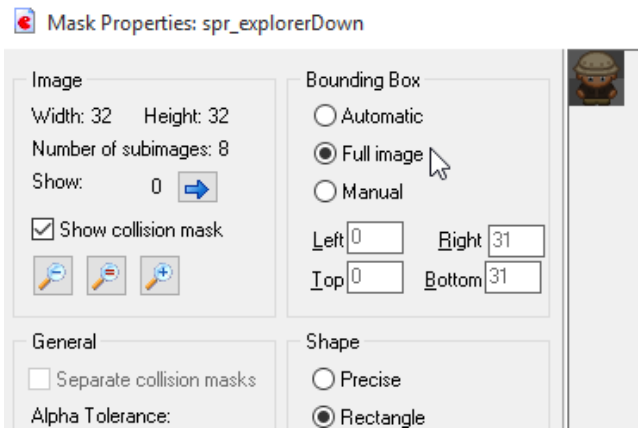
Note that there is a gray line that shows where the image is being chopped.

Hit OK.

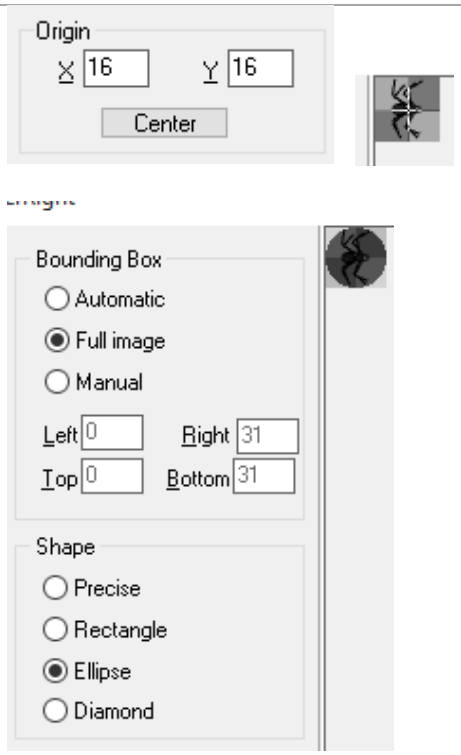
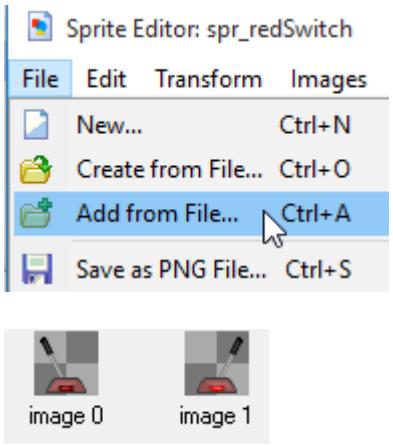


Make sure **precise collision checking** is **off.** Click **Modify Mask** and make sure that the collision mask is using the **Full Image**.
We want this sprite to take up a whole 32x32 square.

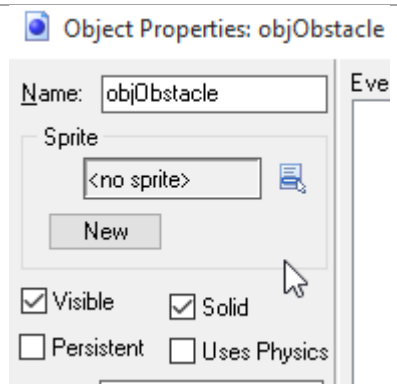
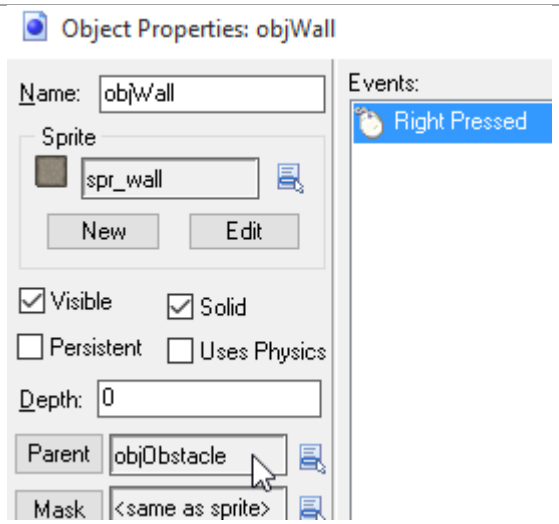
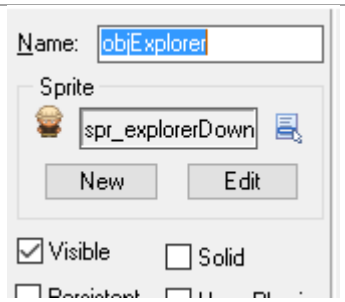
Make sure that the sprite's **Origin** is at **0, 0**
We want to be able to line up the corner of the sprite to the corner of the 32x32 grid – that is why we are leaving the origin at 0, 0 instead of centering it.



Do the same to make **spr_explorerRight, spr_explorerLeft, spr_explorerUp, spr_torch** and **spr_skelleton** from the strips. Note that the skeleton has 10 images and the torch only 3.

Do the same to make **spr_spider** from the **spider_strip8.png** This sprite should be centered. Click **Modify Mask** Set it to use the **Full Image**, but as an **Ellipse** This should make for a circular collision mask that we can spin in circles without getting caught on the wall.	
Bring in **redSwitch_0.png** as a sprite **spr_redSwitch**. Set the **Origin** to 0, 0 and **precision collisions unchecked**. Then **Edit Sprite** and do **File→Add from File** The two frames for the switch are not stored as a strip, so we have to add them by hand. Add **redSwitch_1.png** - it should come in as a second frame in this sprite.	
Do the same for **spr_blueSwitch**	

5 WALLS, EXPLORER & EXIT

Make an **objObstacle** it will not have any sprite or any events. In fact, we will never actually put one in the game. But it will be the parent for anything that is an obstacle bad guys can't walk through.	
Make an **objWall** that is solid and has **objObstacle** as its **Parent** and is **Solid** Optional: A Right Mouse Pressed event that destroys the wall. Allows you to make shortcuts through the maze if you want to while testing.	
Make an **objExplorer**	

Make a **Begin Step** event.

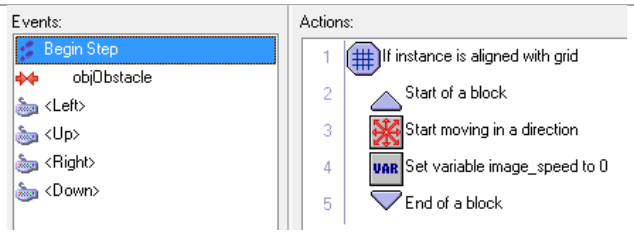
Check to see if the player is aligned with the **32 x 32** grid, if so we want to stop them.
Unless you hold down a key, you move until you get to the next 32x32 square and then stop.

To stop, do **Set Variable image_speed** to **0** (stop playing the animation) and **Start Moving** with **0** speed (stop).

We are using Begin Step because it is executed before any keyboard input is processed. That way when we stop the player in the Begin Step, the key press events can start them moving again before they actually jump to their new positions. Here is the order:

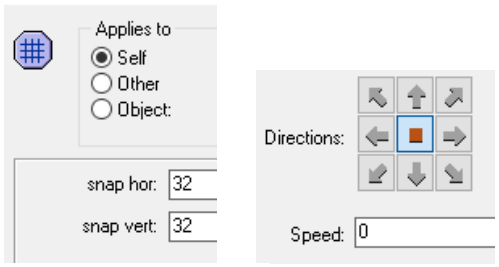
Begin step
Check for input
Step
Move to new location
Collision
End Step

Regular step would happen AFTER the keypress, overriding the movement we tried to set up there. If you try changing Begin Step to Step you will find you can't move!

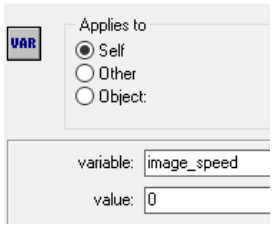


Details:

Check Grid

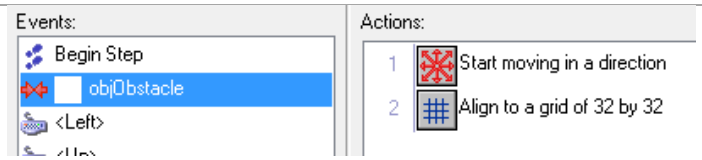


Set Variable

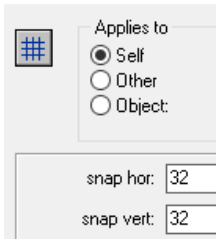


When the player collides with an obstacle they should stop moving (**Move Fixed** with speed **0**) and be forced back on to the 32x32 grid with **Align to Grid.**

Align to Grid forces the object back on to the grid. If the player hits something, we need to make sure they are not stuck partway between two squares. This makes sure that does not happen.



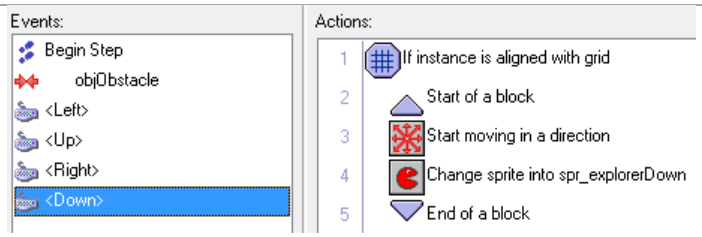
Align to Grid



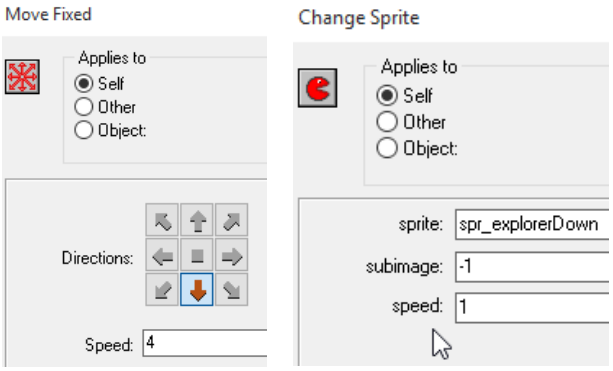
Add a **Keyboard Down** event

First check to see if the player is aligned to the 32x32 grid.
We don't want them to start new movements until they are lined up with the grid. Otherwise the player has to perfectly time everything to avoid catching on corners.

If they are aligned, **Move Fixed** down at speed **4** and change the sprite to be the **down** sprite.
Subimage -1 says "do not change the frame number being shown". That prevents the animation from restarting to the first frame each time over and over as a player moves in one direction.

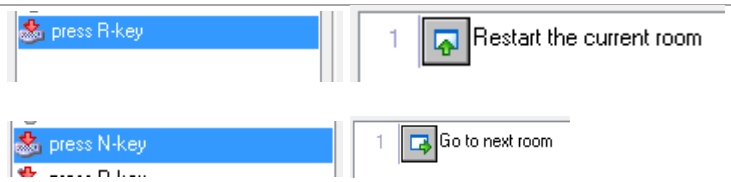


Details:



Do the same for **Keyboard Left**, **Right** and **Up** but change the directions and sprites as appropriate.

Add cheat keys to restart the current room or go to the next room.



Open room1.

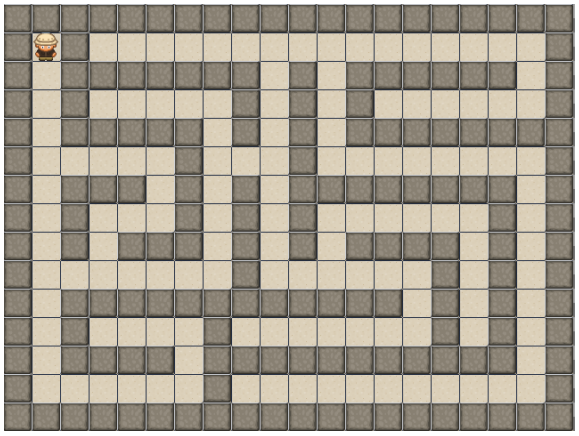
Change the **Snap** to **32** x, **32** y so that you can only place things on the grid.

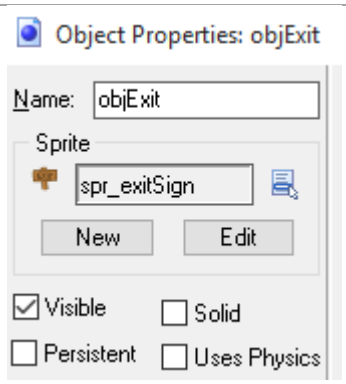
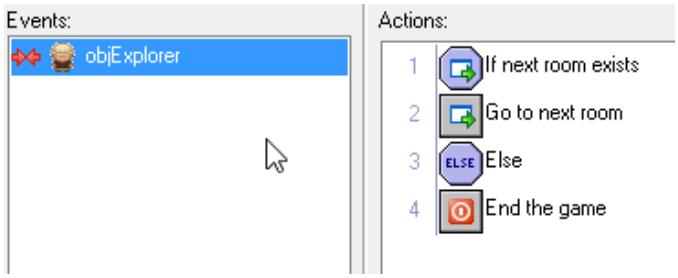
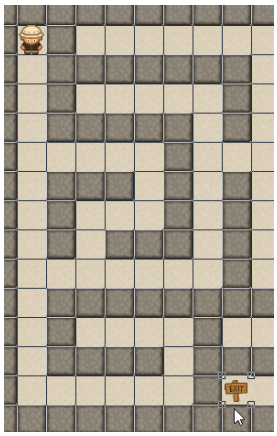
Place walls and the explorer.

Ctrl-Shift Left Click to place items as you move your mouse around (to draw a series of walls).

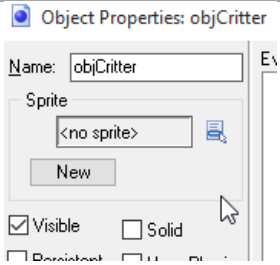
Ctrl-Shift Right Click to erase as you move your mouse around.

(Using this layout may make it easier to exactly follow instructions later…)



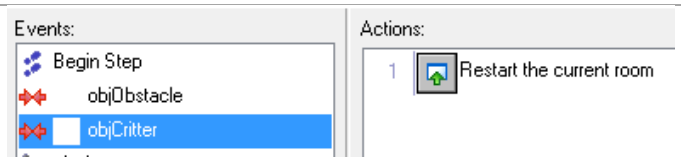
Test your movement code. You should be able to move in all four directions. You should be easily able to turn corners and should not get hung up on anything.	
Add an **objExit** When the player hits the sign, check to see if there is another room. If so, Go to it, otherwise end the game. Place an exit in the room	  

6 SKELETONS & PATHS

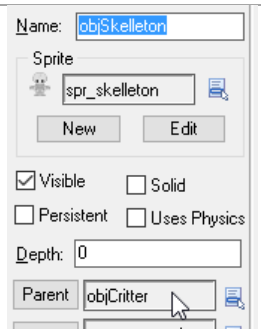
Make an **objCritter** This will be the parent for the creatures that can kill the player.	

Go back to **objPlayer** and add a collision with **objCritter** that restarts the room.

We could put this in critter's collide with player… either one would work fine.

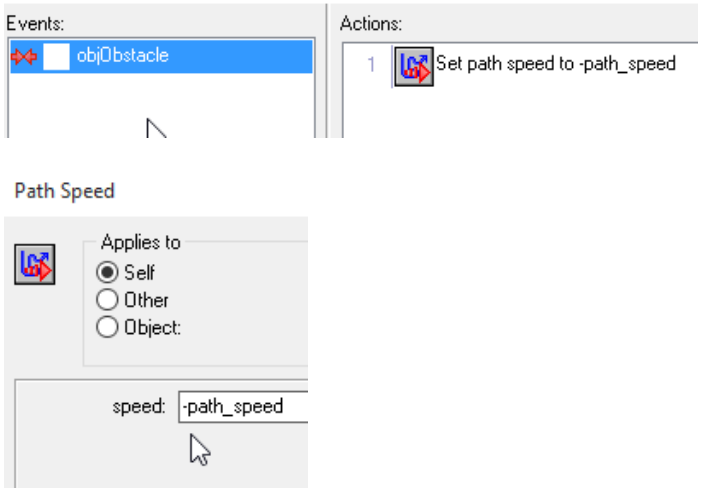


Now make **objSkeleton** with **objCritter** as its parent.

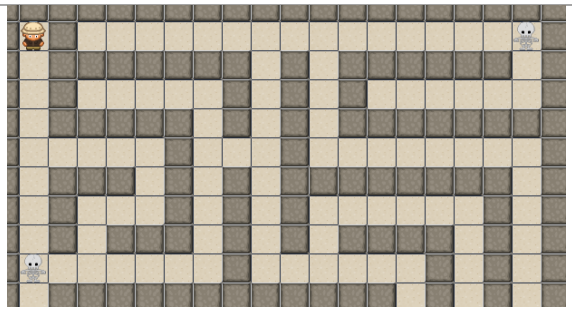


Skelletons will follow a path through the level. When skeleton **collides** with an **objObstacle**, we want it to reverse its movement along its path. Use **Set Path Speed** and for the speed type **-path_speed**

path_speed is the built in variable indicating how fast an object is moving on its path. If path speed is 4, the object moves 4 pixels per step on its path. Setting path_speed to –path_speed change 4 into -4 so not it moves backwards at the same rate it was going forward. If it hits another wall, it will change -4 back to 4 and start going forward again.



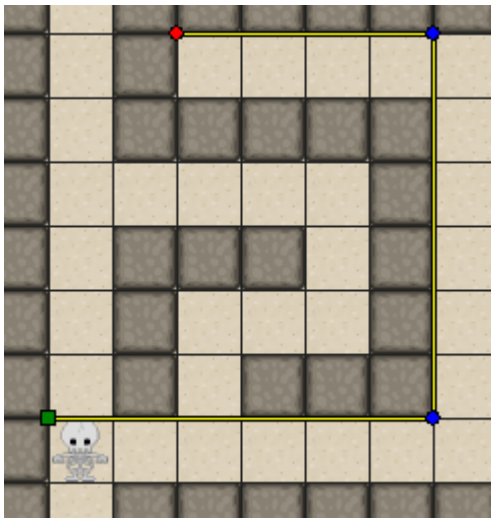
Put two skelletons in the room



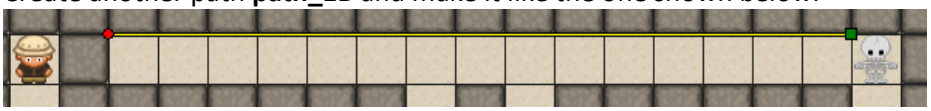
Create a path **path_1A**

Set the snap to 32x32 and then select room1 to use as the background.

You then should see the room to help layout your path. Draw the path as shown to the right. Note that the upper left corner of the skeleton is going to follow the path – make sure that anywhere along the yellow line you place the skeleton it would not be in a wall.



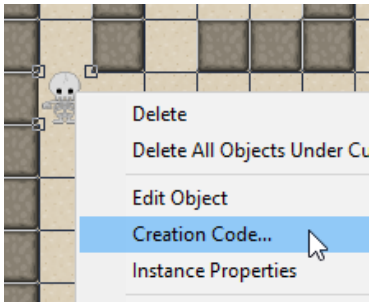
Create another path **path_1B** and make it like the one shown below.



Now we need to make one skeleton follow one path and the other follow the other path. We could make two different skeleton objects (skeletonPath1A and skeletonPath1B) and use their Create events to set things up, but that sounds like it would get out of control fast.

Instead we can use instance **Creation Code** to make a particular instance do something special.

Go to the **room1** editor (not Path editor). Right click on the first skeleton and chose **Creation Code**



Type (or copy/paste) the following into the text box:

```
path_start(path_1A, 4, 3, true);
```

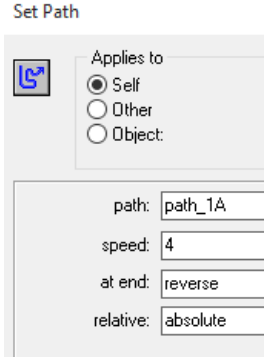
Note that spelling and capitalization must be exact. If you did not name your path_1A exactly "path_1A" you will need to change this.

It says "make this instance follow path1A at speed 4".

*The **3** indicates we want to reverse along the same path once we get to the end. (You can find the other options in the Gamemaker Manual on path_start)*

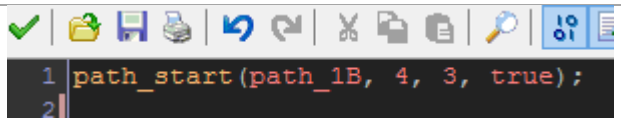
The "true" means "yes, do this path in absolute location within the room". "false" would say "do this path in relative location based on where the object is". It is equivalent to the drag and drop Set Path block shown to the right. But Creation Code can't be drag and drop – it must be text based code, so we have to use the typed version.

What that means (You DO NOT actually use this)



The other skeleton (upper right corner) should have creation code to start it on path 1B:

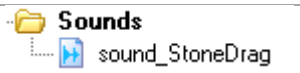
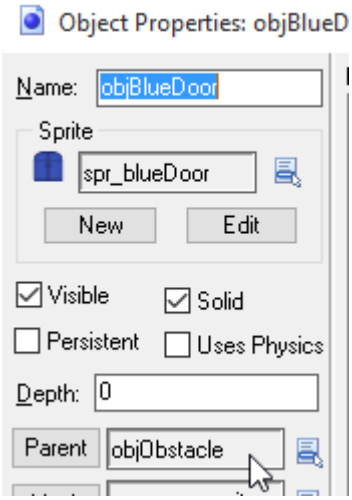
```
path_start(path_1B, 4, 3, true);
```



Try the game make sure the skeletons go back and forth on their paths. Hitting a skeleton should restart the level.

If a skeleton ends up teleporting back and forth after doing its path you have discovered a bug in gamemaker – this sometimes happens at the end of paths. The easiest fix is to change the length of the path – make it one block shorter or longer and you should be set. (OK to make it continue into a block at the end… skeleton will reverse when it hits the block).

7 DOORS AND SWITCHES

Add the sound file **stoneDrag1.wav** as a Sound resource called **sound_StoneDrag**	
Make **objBlueDoor** with **objObstacle** as its **parent** It should be **solid** We don't need to add any extra code – being an obstacle means things already collide with it.	

Make an objRedDoor using same settings.

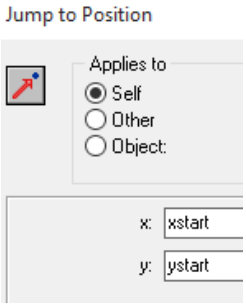
It will “close” automatically
Add **Alarm 0** and make it **Check Empty xstart, ystart**

If it is empty at that location (i.e. player is not standing in the doorway) **Jump to Position xstart**, **ystart**

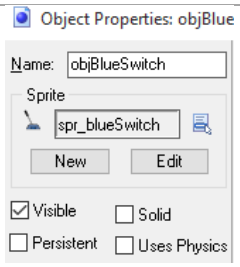
Otherwise, reset **Alarm 0** for **1** step so we check again next step and see if the area is clear yet so the door can close.

xstart and ystart are the location the object was in at when the room started – you can always use them to reset an object

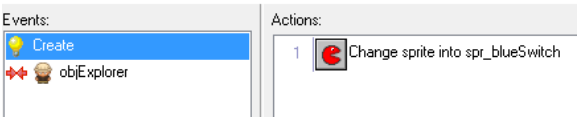
Details:



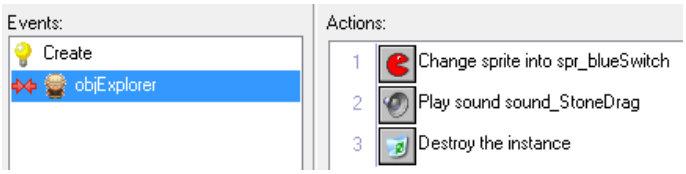
Make an **objBlueSwitch**



Create should set the sprite blue switch sprite but make the image **speed 0** so it is not animated.

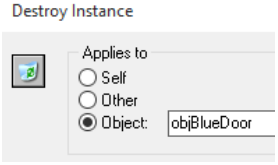


On **collision** with **objExplorer**:
Set the sprite to **spr_blueSwitch** with **subimage 1** and **speed 0** (to show that it has been switched)
Play sound **sound_StoneDrag**; **looping**: **false**
For all **objBlueDoor**: destroy the instance



The sound is important to let the player know something big happened. Later on they may not be able to see the door they opened.

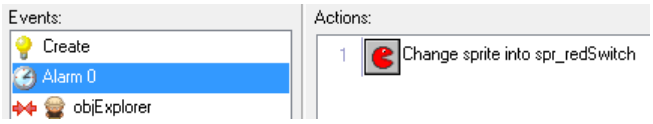
Make sure to destroy blue doors (this destroys all of them)

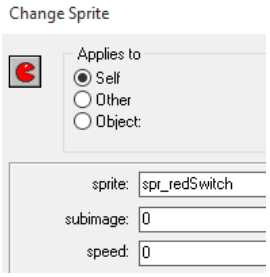


Make an **objRedSwitch**

It's create event should be just like the blue switch (set the sprite to spr_redSwitch but set speed to 0).

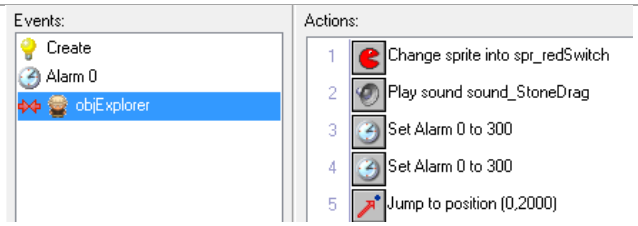
Alarm 0 should set the sprite back to **subimage 0** and **speed 0** to reset it



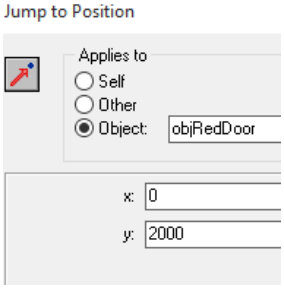


When the explorer hits this switch, we will move the red door offscreen (but not destroy it). We will set alarms for the switch and red door to reset them after 10 seconds (300 steps).

- set the sprite to **spr_redSwitch** with **subimage 1** and **speed 0**
- play sound **sound_StoneDrag**; looping: **false**
- set **Alarm 0** to **300**
- for all **objRedDoor:** set **Alarm 0** to **300**
- for all **objRedDoor:** jump relative to position **(0,2000)**

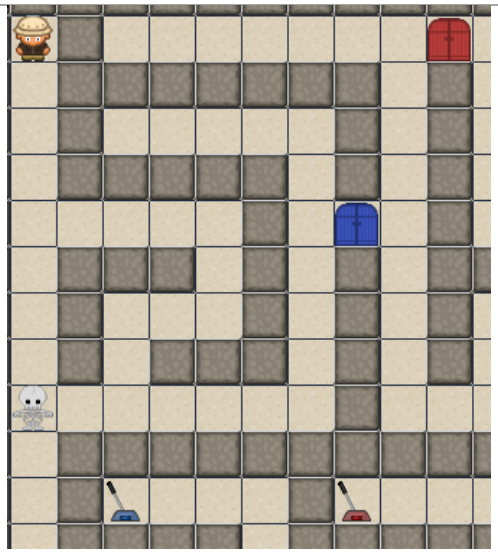


Make sure last two apply to objRedDoor:



Place the switches and doors and test them out.

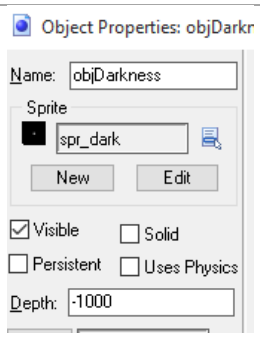
Make sure the red door returns after 10 seconds.



8 DARKNESS

Make **objDarkness**

Give it a depth of -1000 so it appears on top of everything else (objects with a lower depth value are drawn on top of those with a higher depth value).

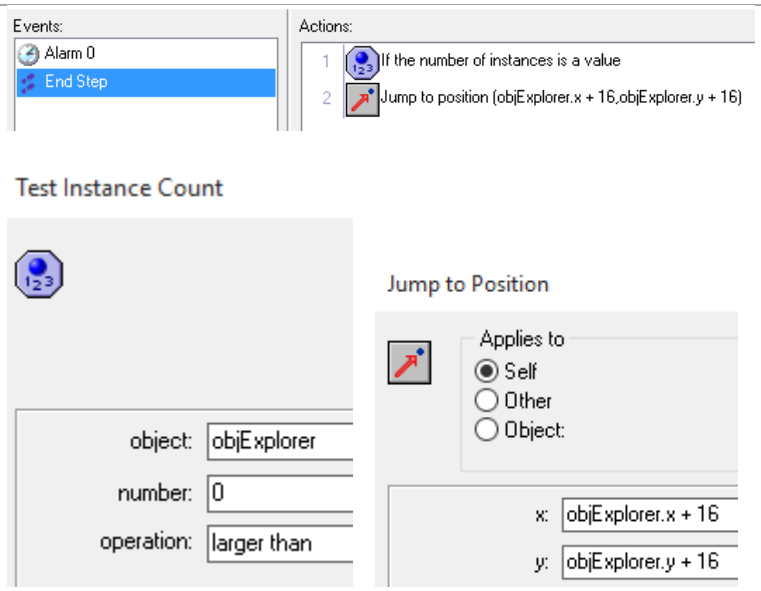


Start with an End Step event – it should check to make sure there is an explorer on the screen. If so, jump to their location.

If **Count Instances objExplorer larger** than **0** **Jump** to x: **objExplorer.x + 16** y: **objExplorer.y + 16**

Make sure your spelling here matches your spelling for the name of objExplorer! Otherwise your game will crash and tell you it could not "get objExplorer.x".

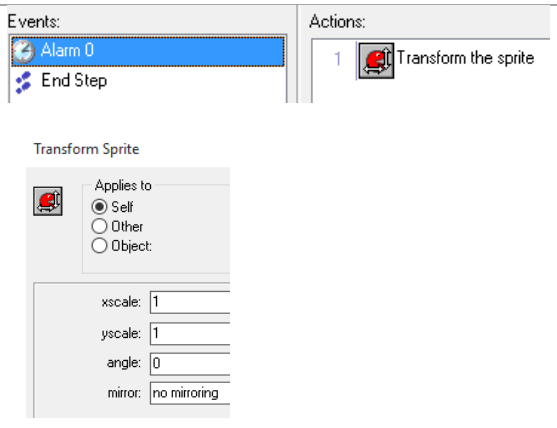
We add 16 to the x and y of objExplorer because we want to find the visual center of the sprite. The origin of the sprite is in the upper left (0, 0) – to get to the middle we need to add 16 to both dimensions.



Add an Alarm 0 which will handle "resetting" the darkness to its normal state.

Alarm 0:

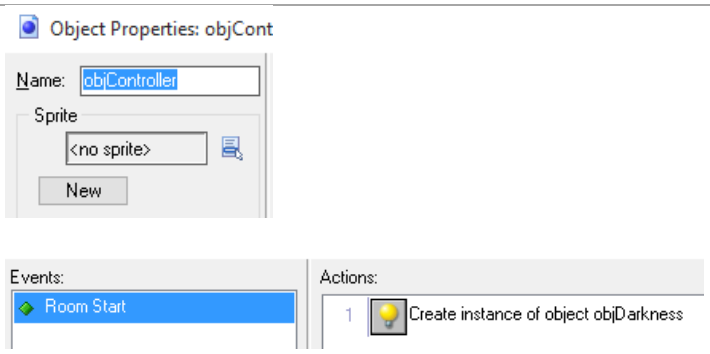
- **Scale** the sprite with **1** in the xdir, **1** in the ydir, rotate over **0**, and **no mirroring**



Add an **objController**

Its only task is on **Room Start** to **create** an **objDarkness** (at any coordinates… the darkness automatically moves to the player)

If you put a darkness in the room it hides everything else in the level editor… this prevents that.



Place a controller in the room.

You should end up with the darkness sprite following the player. It is big enough to cover up the entire room but has a transparent hole in the middle to show the area around the player.

If everything is black, verify that spr_dark is centered (that the origin is in the transparent part).

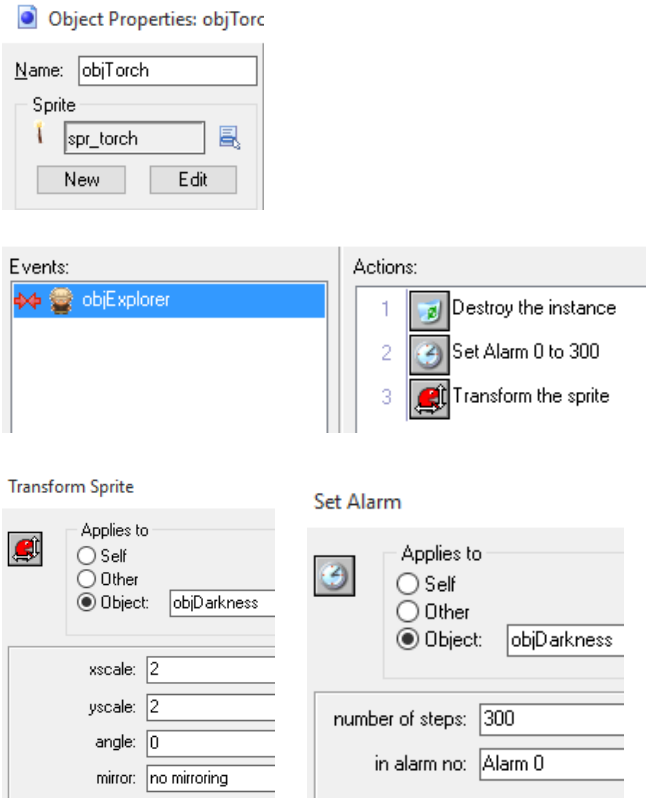
Now add **objTorch**. Picking it up will expand your light for a few seconds.

To do so, add a collision with objExplorer:

- Destroy the torch
- For all **objDarkness**: set **Alarm 0** to **300**
- For all **objDarkness**: scale the sprite with xscale **2** and yscale **2**

Doubling the size of the dark sprite will double the transparent part in the middle as well!

Setting the alarm for the objDarkness makes sure it resets its size after 10 seconds.

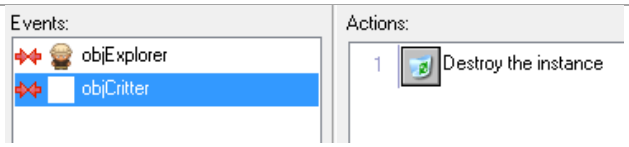


Place a torch in the room and make sure it works.

9 ROCKS

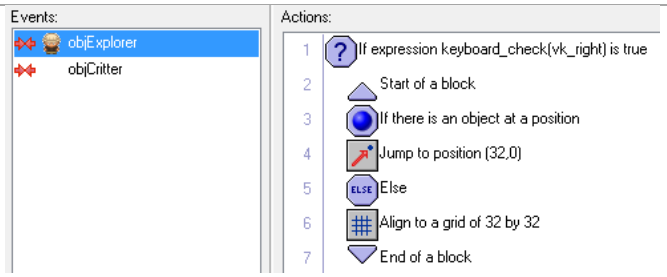
Make an **objRock**
It should be **solid** and should have **objObstacle** as its **parent**.

Add a **collision** with **objCritter** that **Destroys Other** (the critter).
Pushing a rock into a critter will kill it.



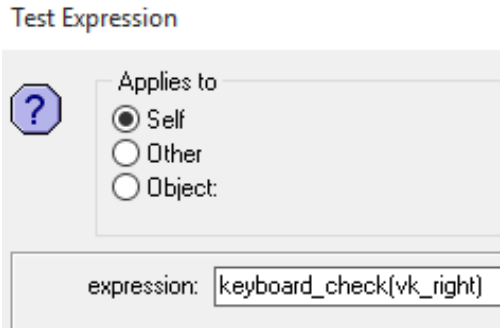
The explorer should be able to push rocks unless there is something blocking them.

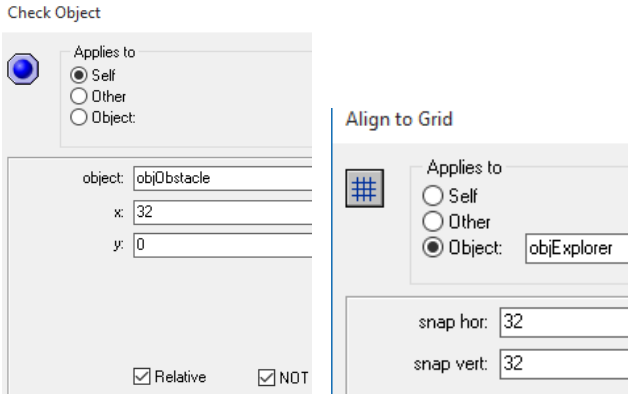
When the explorer hits a block, we will check what direction key is down to determine which way to try to move the rock.



Add **Collision** with **objExplorer**

- If **Test expression keyboard_check(vk_right)** is true
  - If **Check Object** at **relative** position (**32,0**) there is **not** object **objObstacle**
    - **jump relative** to position (**32,0**)
  - else
    - for all **objExplorer**: **align to grid** with cells of **32** by **32** pixels

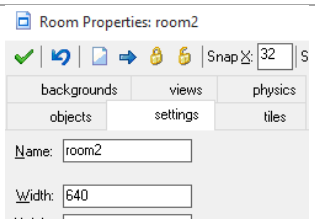




keyboard_check( ) *tests to see if a key is down; make sure to use parentheses:* (*not brackets:* [
vk_right *is how you say "right key"*

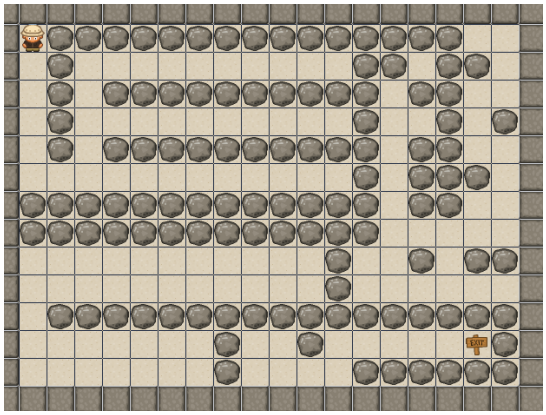
For more info see:
Keyboard Input in Gamemaker Manual

Make a room2



Make a maze of rocks for the player to maneuver through.

Test and make sure you can move around and push rocks (only to the right for now) and that you cannot push rocks with something solid behind them.

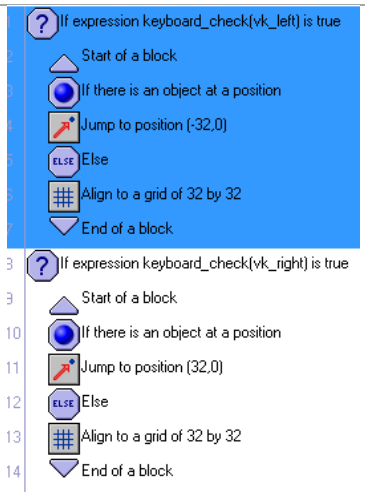


Once you have right push working, time to add the rest of the directions.

Go back to rock's collision with person. We want to add three more sets of code just like the "if right key is pressed" to handle the other directions.

The screenshot to the right shows the new code for left highlighted in blue. You would also need a copy of the blue block for up and another for down.

Hint: Ctrl-Click can select multiple actions so you can copy and paste a whole block of them. Just make sure to change the details after copy-pasting!!!



The differences:

Left:

- Use **vk_left** in the **keyboard_check**
- **x** in the Check Object and Jump is **-32**

Up:

- Use **vk_up** in the **keyboard_check**
- **x** in the Check Object and Jump is **0** but **y** is **-32**

Down:

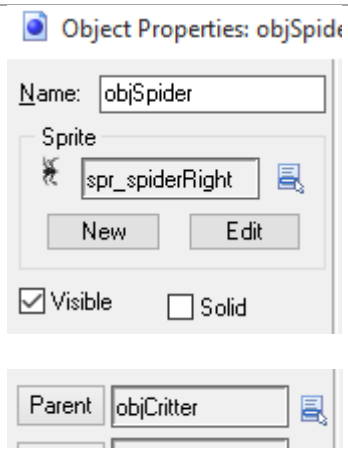
- Use **vk_down** in the **keyboard_check**
- **x** in the Check Object and Jump is 0 but **y** is **32**

Test and make sure you can push rocks in each direction.

10 SPIDERS

Add an **objSpider** that has **objCritter** as its parent.

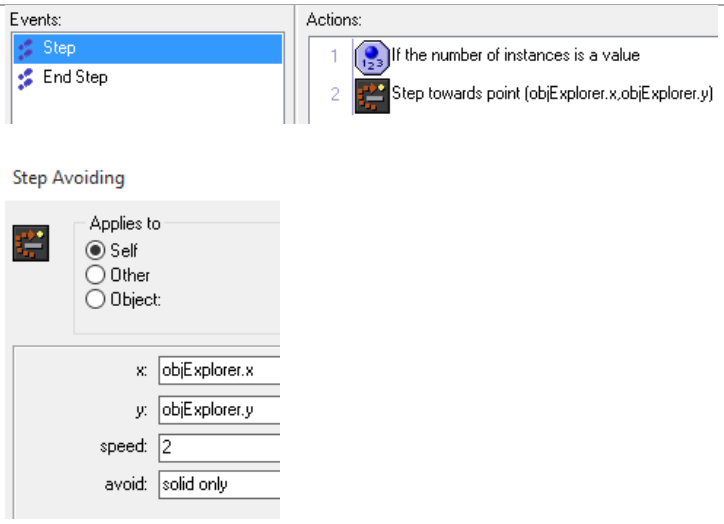
Spiders will try to chase the player down through the maze



The Step event should try check to make sure there is an explorer and if so try to move towards him.

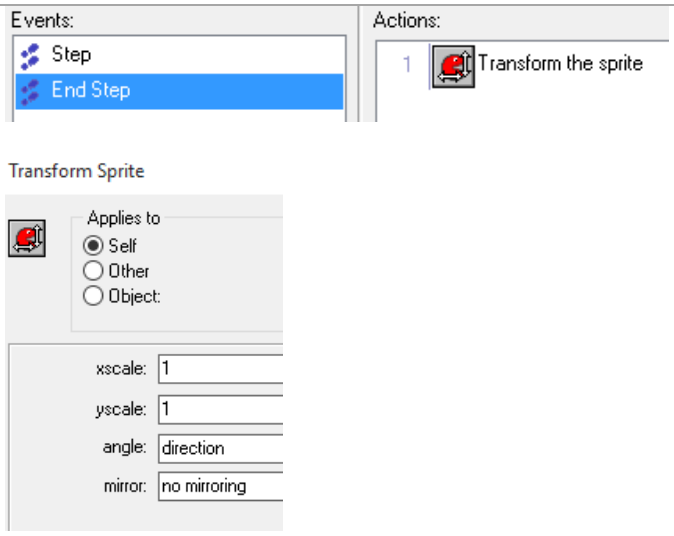
If **Check Count** for **objExplorer** is **larger** than **0** Perform a **step towards** position (**objExplorer.x, objExplorer.y)** with speed **2** avoiding **solid only**

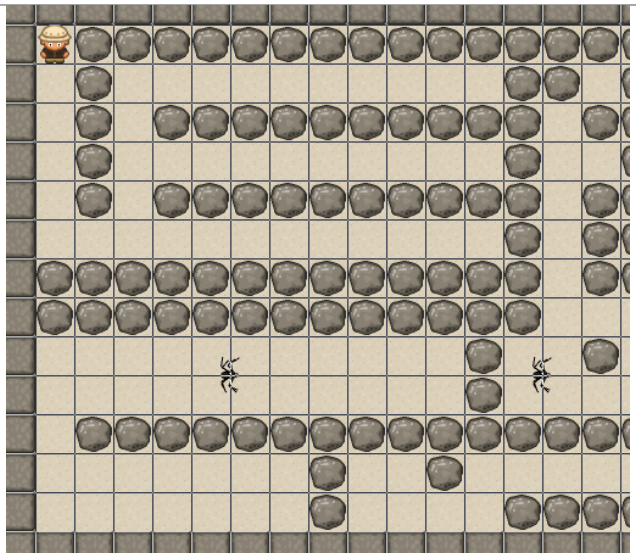
Step avoiding tries to move to reach the indicated point and will navigate around obstacles. It does not do detailed path planning… it will have trouble getting through a maze, but will seek out the player.



In **End Step**, do a **transform sprite** to set its **angle** to **direction**

This uses the built in variable direction to rotate the sprite to face whatever direction the object just moved.



Add a few spiders to the room. Note that because of their sprite's centering they will want to snap to grid corners. If you put them partly inside walls they will be instantly crushed! Test out that they move around and that you can smash them with pushed rocks.	

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.

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)
- Shooting
- Different goals within a level (find 4 gems to open the exit)
- 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)
- Secret doors
- Traps