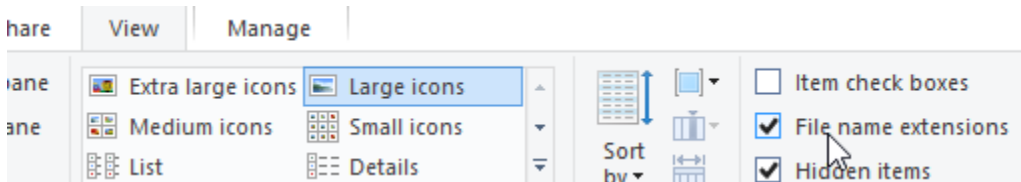


1 Shooting Gallery Guide

2 SETUP

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

In the file explorer, go to the View tab and check File name extensions. This will show you the three letter “extension” of file names that are normally hidden from you. When I refer to a file name I will use the full name like “playerPicture.png” not the simplified one that you would normally see (just “playerPicture”).

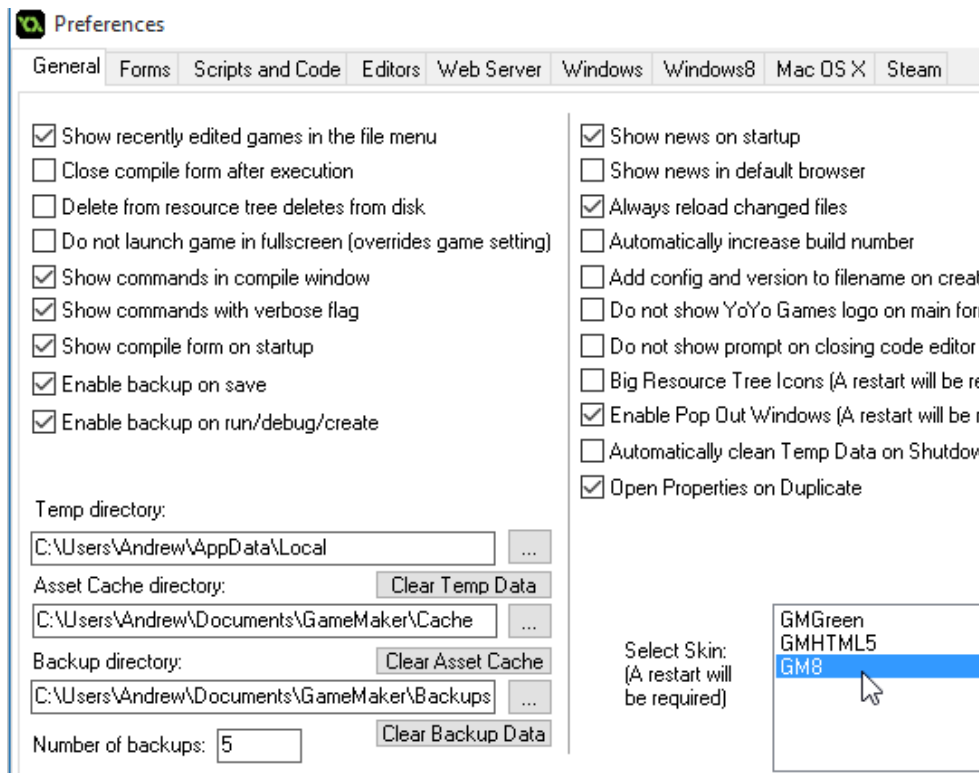


Open Gamemaker and start a New Project. Name it **ShootingGallery**. Make sure to pay attention to where you save the project – I would recommend you make a folder in your Documents folder for all your games.

You can close the **News** window in the center of the main window.

Go to **File menu** then **Preferences**

Select the GM8 Skin – this will replace the default green and black coloring with more colorful icons that will match my screenshots and be MUCH easier to tell apart.

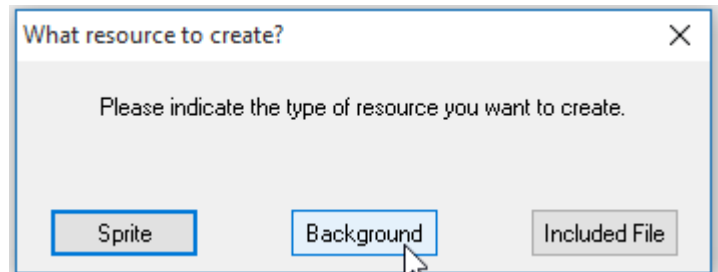
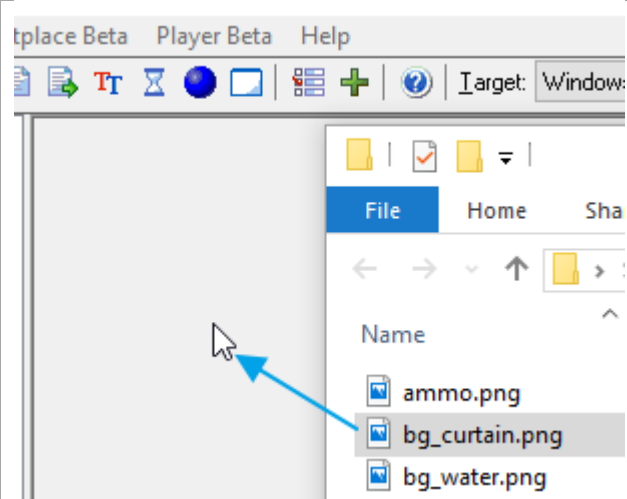


Close Gamemaker Studio then reopen it.

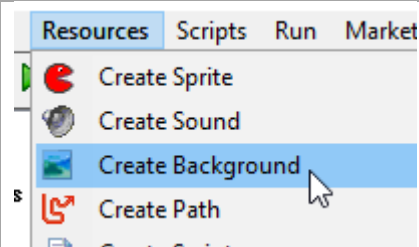
3 BACKGROUNDS

First load the images we will use as backgrounds. The easiest way is to drag from the file explorer into Gamemaker and then chose **Background**.

Do this for **bg_water**, **bg_wood** and **bg_curtain**

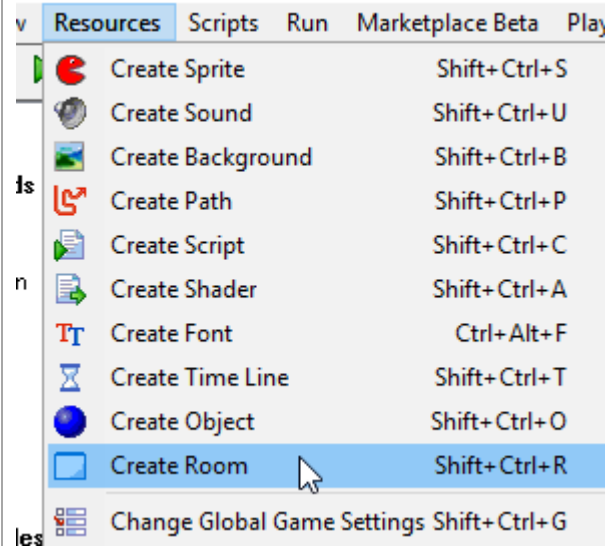


Instead of dragging from explorer, you can also **Create Background** in Gamemaker. Using the **Load Background** button will then allow you to pick the image file to use.



4 FIRST ROOM

From the **Resources** menu, **Create Room**



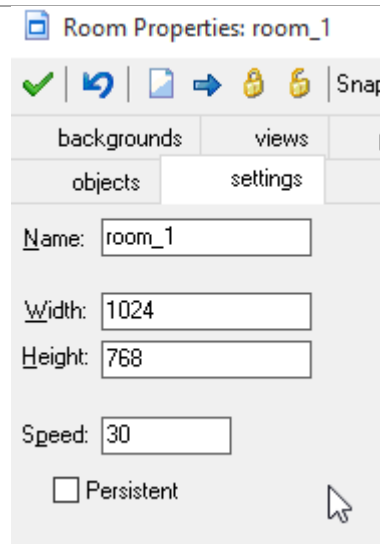
Leave the default **Width** and **Height**. If you later want to make a room that is bigger or smaller, you would make that change here.

Also leave the default speed. Speed controls how many times per second the game updates each object. (This is NOT FPS – while there are times when cranking it up can make this smoother, you can't just change this number to make your game run at 120 FPS or something).

Set the name to **room_1** While technically it work to leave it as room0, it is nice to give everything a meaningful name.

Tips on names

- Never use spaces or weird characters in file names. Instead of a space, use _ like this: *one_two* or smash the words together like: *oneTwo*
- It is VERY handy to start with the type of thing in the name:
 - *object_* for Objects
 - *room_* for Rooms
 - *sprite_* for Sprites
- Capitalization counts. *object_thing* is NOT the same as *object_Thing*
- If you use different names than the instructions, you will have to be careful to modify the instructions to match your names!

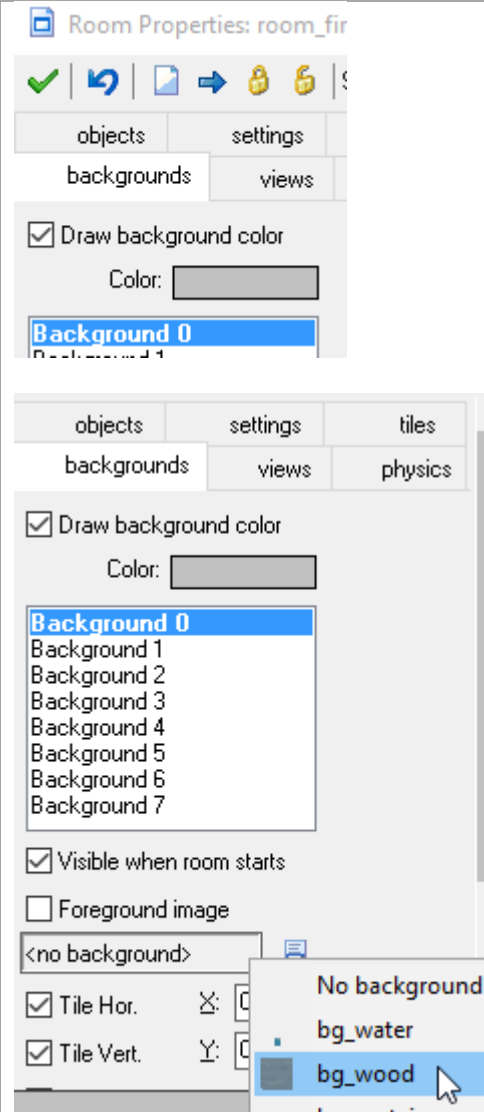


Go to the **Backgrounds** tab.

Select **Background 0**

Then click the <no background> drop down and select **bg_wood**.

Notice it is set to Tile Horizontally and Vertically. That means it gets repeated across the whole room.

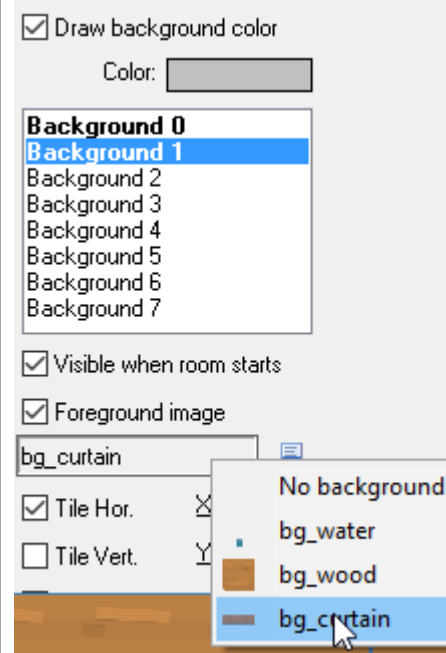


Select **Background 1** and set it to be **bg_curtain**.

It will automatically be on top of Background 0.

Uncheck **Tile Vert.**. Leave **Tile Horizontally** checked so it is repeated across the top but repeated down the rest of the room.

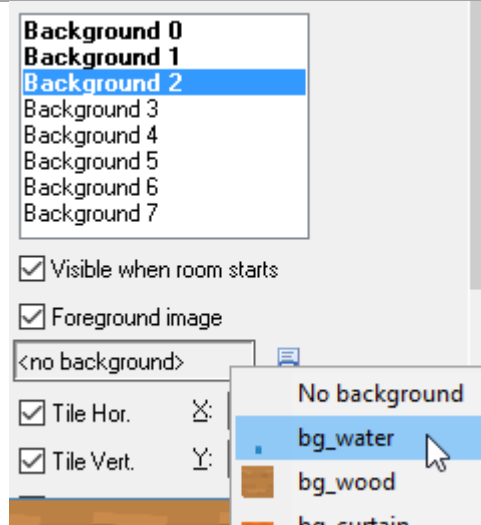
Set it to **Foreground Image**. That will make it always appear in front of all the objects in the game. We will want the curtain to show in front of any targets that are in the game. (Like how the water covers the duck in the next step)



Select **Background 2** and set it to be **bg_water**.

Set it to **Foreground Image** also. We will want the water to show in front of any targets that are in the game as shown in the bottom picture to the right.

Note that the bg_water image has a lot of extra empty space at the top – it is already 768 pixels tall (same as the room). That is why it appears at the bottom of the room instead of the top. There is no easy magic way to say “put this background image at the bottom of the screen”.



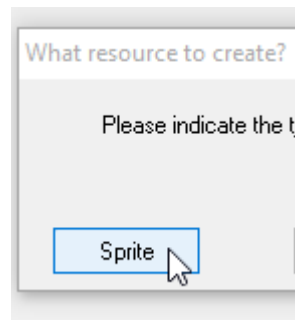
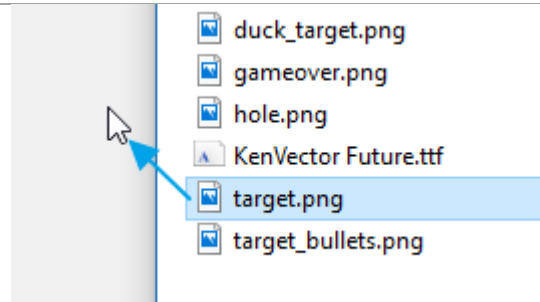
Test your game by hitting the Run button. There is nothing to do yet, but the room should look like the picture shown.

If something is wrong, go back and fix it now!

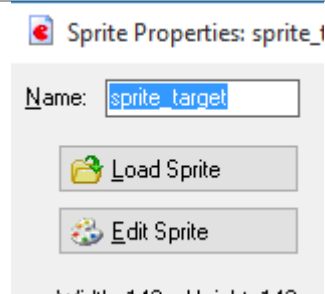


5 SPRITES

Just like with the backgrounds, drag in the **target.png** image but this time chose Sprite

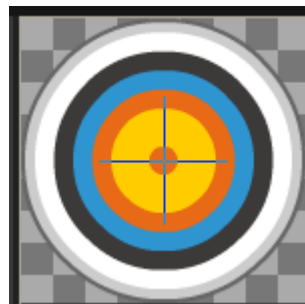
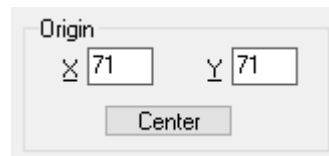


Name is **sprite_target**



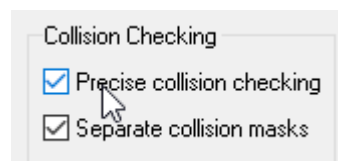
The **Origin** is the spot on the sprite that is considered to be its “center”. All things are measured from that location. In some games, where you make the origin of sprites doesn’t matter much. In others it is critical.

Click the **Center** button to automatically center the Origin. Notice that the preview image now shows a crosshairs in the center – that is where the Origin is. Any time we measure distances to this sprite we will measure to that center point. If we rotate this sprite it will rotate around that point.



Finally, check **Precise collision checking**.

This will make it so that any check to see if we hit the target will consider only the pixels to count as hits. The grey checkerboard around the outside is showing us there the image is transparent – they will not count as hits. If Precise collision checking was off, we would be able to hit this target by shooting those corners!



Bring in **target_bullets.png** as **sprite_target_bullets** in exactly the same way.

Bring in the **duck.png** file as **sprite_duck**

0, 0 is the upper left corner of the image. Since the image is 114x109, the middle of the bottom is at x: 57 and y: 109.


Set the Origin to that location (57, 109)

The Origin crosshairs should show at the bottom center of the duck.

Turn on **Precise Collision Checking**

***X GETS BIGGER GOING TO THE RIGHT
Y GETS BIGGER GOING DOWN – IT IS “BACKWARDS”
FROM WHAT YOU LEARNED IN MATH. THIS IS PRETTY
COMMON IN COMPUTER GRAPHICS.***

Name:

 Load Sprite

 Edit Sprite

Width: 114 Height: 109

Number of subimages: 1

Origin

X

Y



Bring in **duck_target.png** as **sprite_duck_target**

Set the Origin just like the normal duck.

Turn on **Precise collisions**

Click **Modify Mask** to change what counts as a “hit” on this sprite.

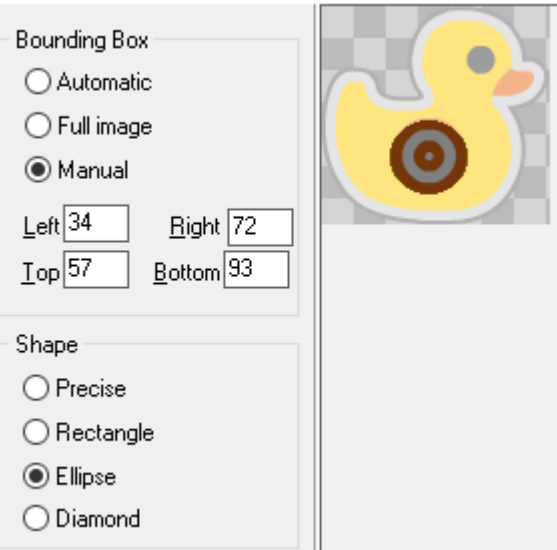
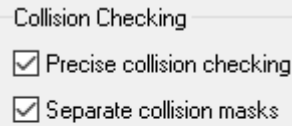
Change the **Bounding Box** to **Manual**

This lets us define the region that is a hit

Change the **Shape** to **Ellipse**

Play with the Left, Right Top and Bottom numbers until you get a circle that just covers the target on the duck.

Now you can only hit this duck by hitting the target, not by hitting the yellow part.



Bring in **sprite_crosshair** and **sprite_hole**

Center the **Origin** of both. (We want to “shoot” in the center of the crosshairs)

They will not be colliding, so don’t worry about Precise Collisions

Bring in the **sprite_gameover** don’t worry about precise.

Do center the **Origin**, we will want to line up text to its middle.

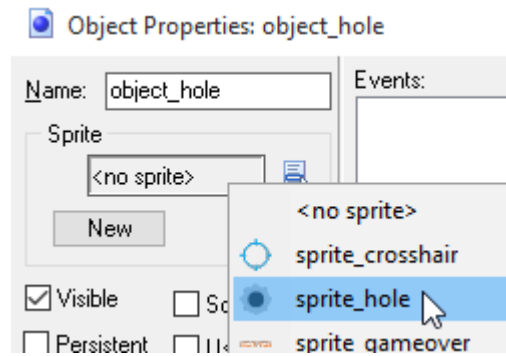
6 HOLES

From the **Resources** menu, **Create Object**

Set the name to object_hole

Object names are the most important thing to keep straight, if you get lazy about naming them it will make it MUCH easier to make a hard to find mistake in your code

These will be the bullet holes that appear as the player shoots.



Click **Add Event**

Click **Create**

This event fires when the object appears. We want to start a timer that will make the hole disappear so the screen does not fill with them.

Click the **Main2** events tab on the right side of the window.

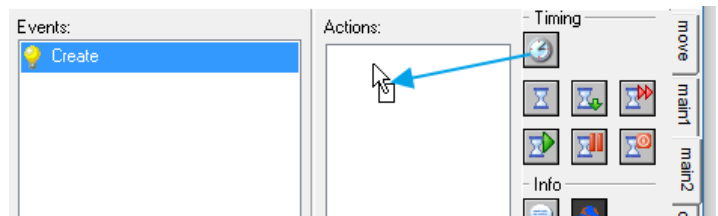
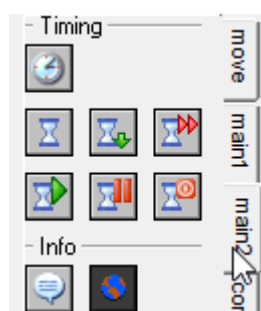
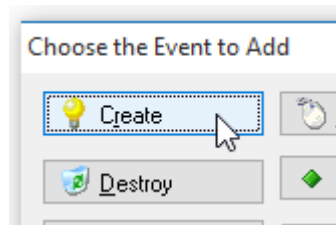
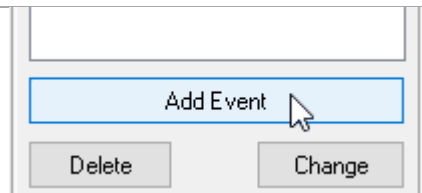
Drag in the **Alarm** action  to the **Actions** window

WHEN YOU ARE NOT SURE WHERE TO FIND AN EVENT OR WHAT IT LOOKS LIKE, YOU CAN LOOK IT UP IN THE ACTION GUIDES ON MY WEBSITE!

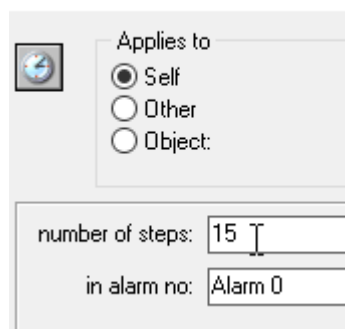
Set **Alarm 0** to **15** steps. Our room uses a speed of 30 steps per second, so 15 steps is half a second.

This will cause Alarm 0 for this object to go off in 0.5 seconds, allowing us to make something happen at that time.

Every object has its own alarms and has 12 different ones you can select (to have multiple delayed sets of actions).



Set Alarm



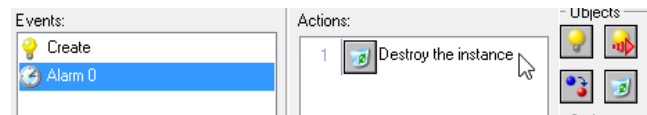
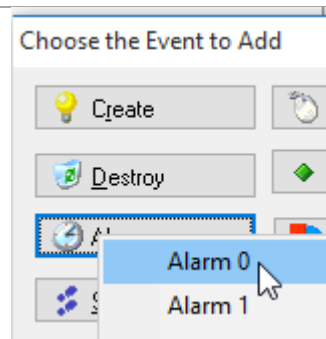
Click **Add Event** again

Select **Alarm** → **Alarm 0**

This is the event that happens when the alarm 0 goes off – whatever actions we put in this event will happen then.

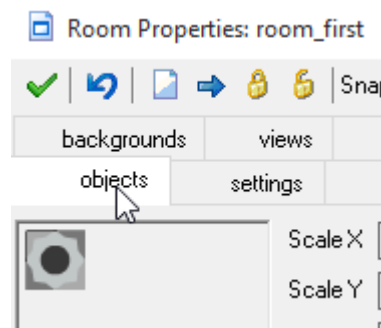
Select the main 1 tab

Drag in **Destroy** to make this hole destroy itself after half a second



Test the holes.

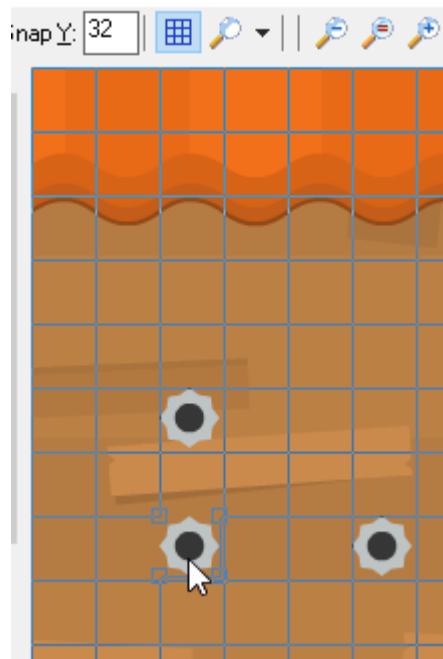
Open the **room_1** again. Click the **Objects** tab to place objects.



Click a couple of times in the room to place some holes.

Run the game – they should be visible briefly before disappearing.

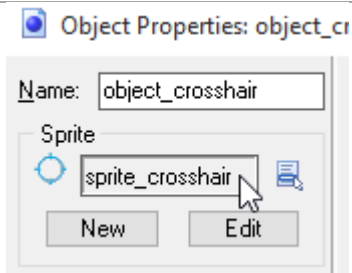
If there is a problem, go double check your work. If it works, go back to the room and delete the holes by CTRL-Right Clicking on them. The holes will not start in the room normally.



7 CROSSHAIRS

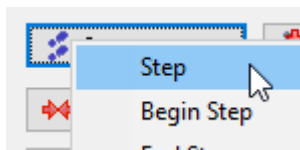
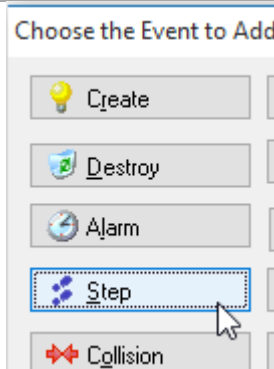
Create a new object for the crosshairs called **object_crosshair**

Select the matching sprite.



Add a **Step** event – chose **Step** from the list of options

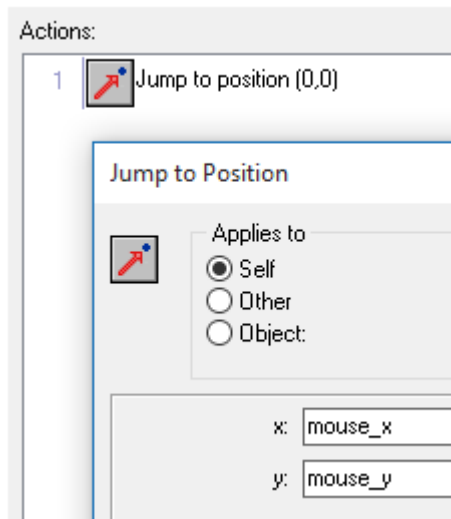
This event fires every step of the game. Any time you want something to get updated all the time, it should go in a step event. Just watch out not to do insanely complex work in your steps or it could bog down your game.



From the **Move** tab of actions bring in **Jump to Position**. This teleports the object to the indicated place.

We want the crosshairs to follow the mouse, so we will use the special variables *mouse_x* and *mouse_y*. Setting the x to *mouse_x* will set the horizontal location of the crosshairs to match the x position of the mouse. Doing the same with the y's will set the vertical position.

Note – how do you learn about things like *mouse_x*? Googling what you want to do can sometimes turn up the answer, or you can check the manual: <http://docs.yoyogames.com/index.html?page=sources%2Fdialogs%2Findex.html>



Add a **Mouse** event

Pick **Global mouse left pressed**

This fires anytime you click anywhere on the screen while the crosshair is in the room. If you used the normal Left pressed (not global) it would only happen when you click on the crosshair. That would probably work too, but if you always want a click to happen, even if you are not over the object, use global. Use the normal mouse click for things that you want to only happen when the mouse is hovering over.

Any **pressed** event happens only once per button click. The **left button** event keeps firing as long as the button is held. If you use left button instead of pressed, the player will have a machine gun that keeps firing while they hold the button.

From the **Main1** actions tab, chose **Create Instance**
Select **object_hole...**

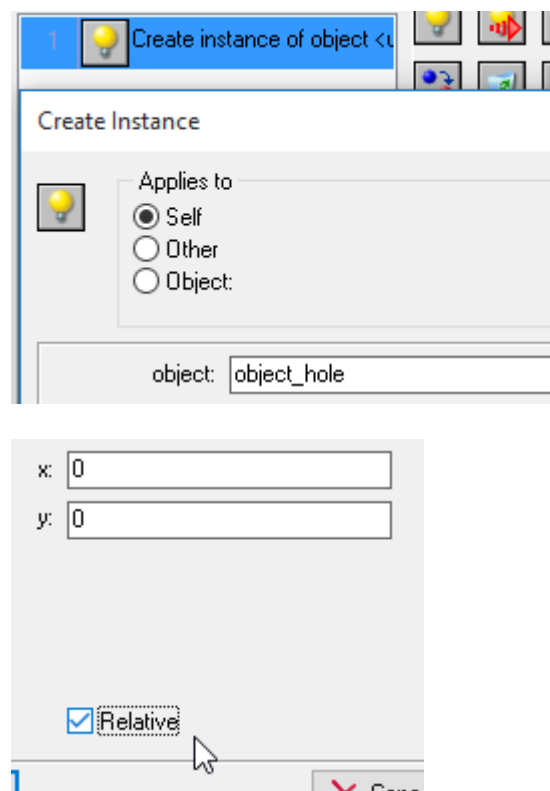
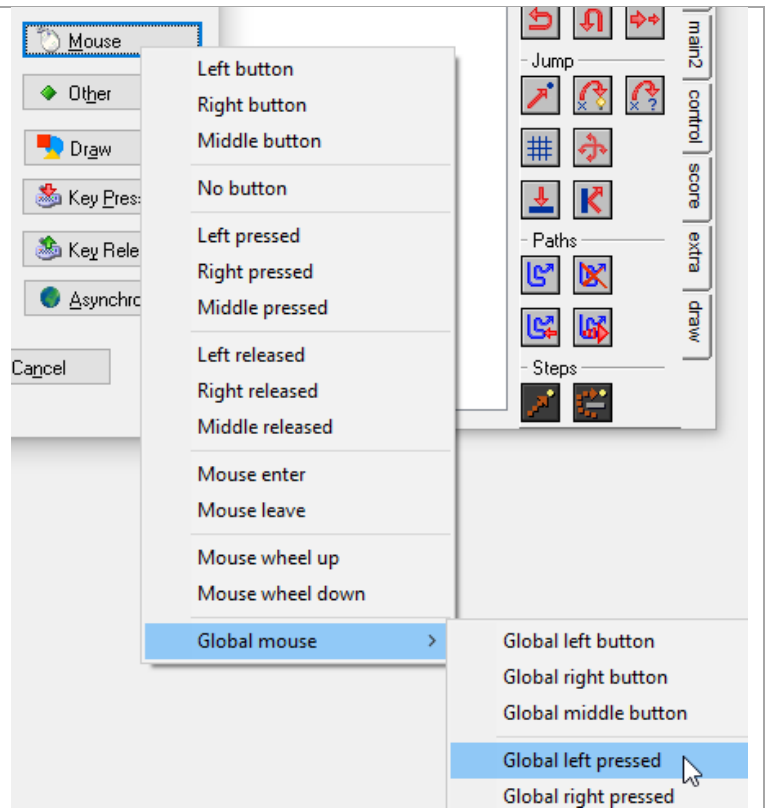
Set the x and y location of the hole to be 0, 0 **relative**

Relative means measure from this object (the crosshairs). So 0, 0 relative means place a hole 0 over and 0 down from the crosshairs.

If you miss the relative checkbox, the holes will always appear in the upper left corner of the screen (0, 0 absolute room coordinates).

IF YOU WANT SOMETHING TO APPEAR IN A LOCATION RELATIVE TO SOMETHING ELSE, USE RELATIVE. IF IT SHOULD APPEAR AT A SET LOCATION IN THE ROOM, DO NOT USE RELATIVE.

Messing up Relative is one of the most common mistakes to make – if something is not appearing in the right location (or at all) – double check if Relative is set right where it is being created.



Add a **Draw GUI** event

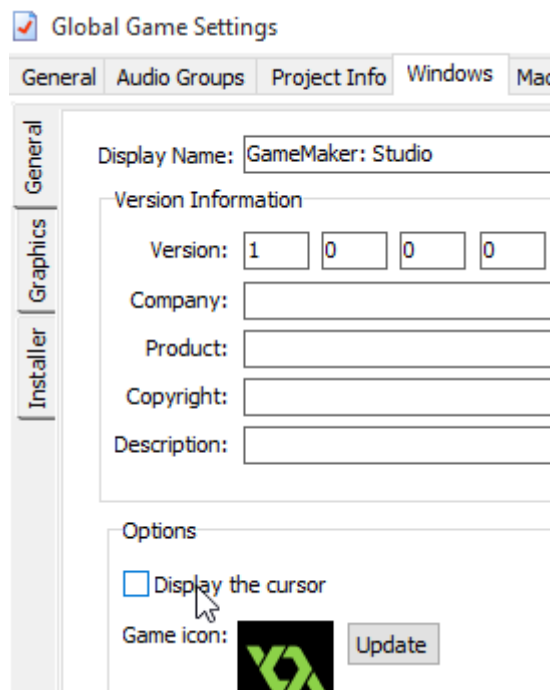
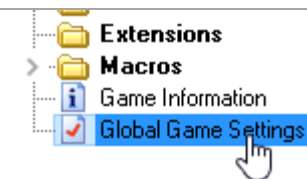
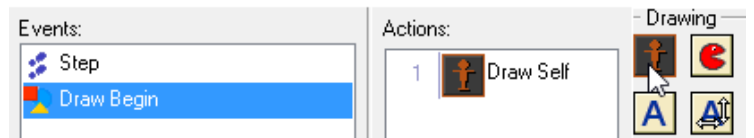
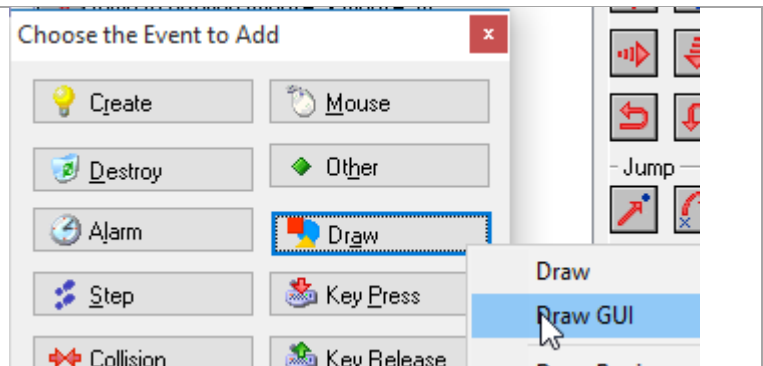
Anytime you want to draw something on the screen, you need a Draw or Draw GUI event. Draw GUI (Graphical User Interface) automatically draws on top of everything else in the game. It is used for drawing things like the score so they do not get covered up.

We are going to use Draw GUI to make sure the crosshairs are always on top.

From the **Draw** tab of actions, drag in **Draw Self**. Now this object does not draw itself in the normal way, it draws itself as a GUI element that is always in front of other things.

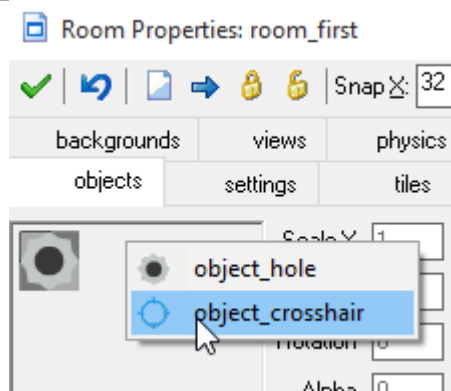
We do not want the normal mouse cursor to appear. So open **Global Game Settings** in the resources window:

On the **Windows** tab, find **Display the Cursor** and uncheck it.



Open the room, go to the objects tab and select the **object_crosshair**

Place one in the room by clicking in the room somewhere – the location does not matter as it will move to the mouse location.



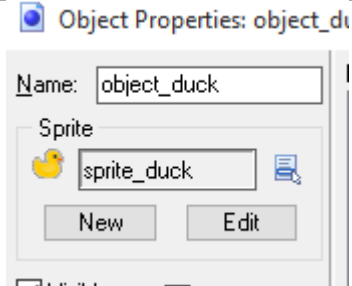
Test your crosshairs. Fire up the game and click all over – bullet holes should appear and slowly disappear.

As always, if there is an issue, go try to fix it before going on.

8 DUCKS

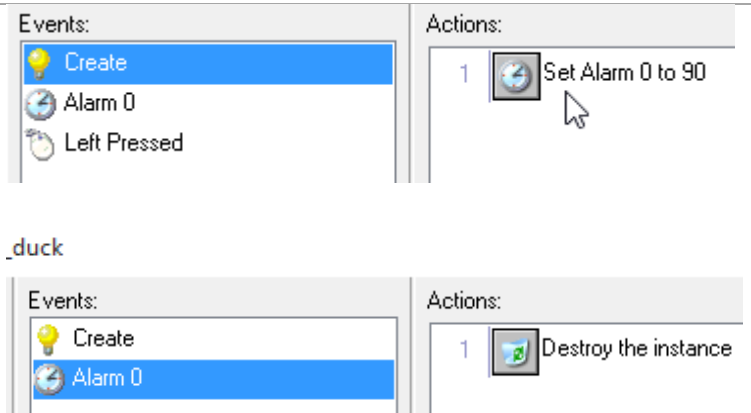
Create **object_duck**

Select the matching sprite.



We want the ducks to only last for a few seconds. Add a **Create** event – set **alarm 0** for **90** steps (3 seconds)

Then in the **alarm 0** event make it **destroy** itself



We want clicking on the duck to “shoot it”

Add a **Left pressed** event (not global – you have to click on it to shoot it)

Add a **Destroy** action and, from the **score** tab of actions, **Set Score**

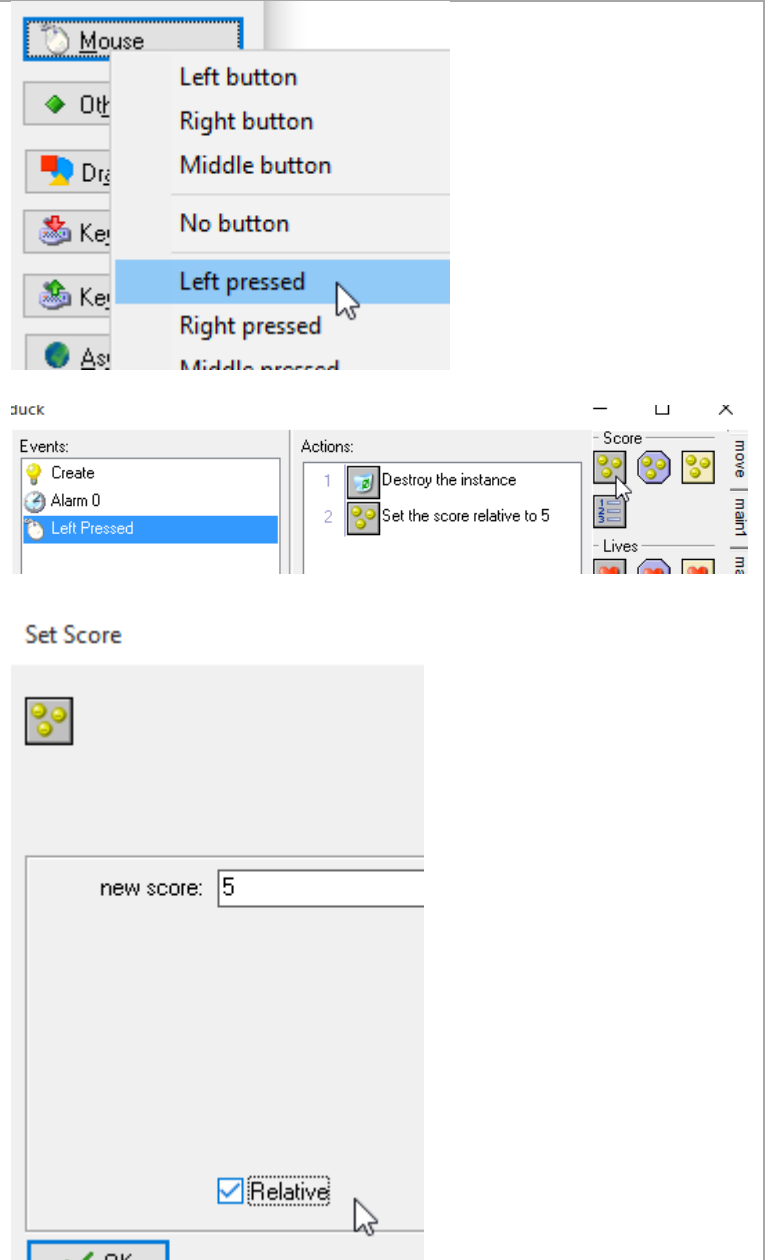
For the score, type 5 and check **relative**

5 relative means “Add 5 to whatever the score is now”

5 not relative means “Set the score to 5”

-3 relative would be “subtract three from the score”

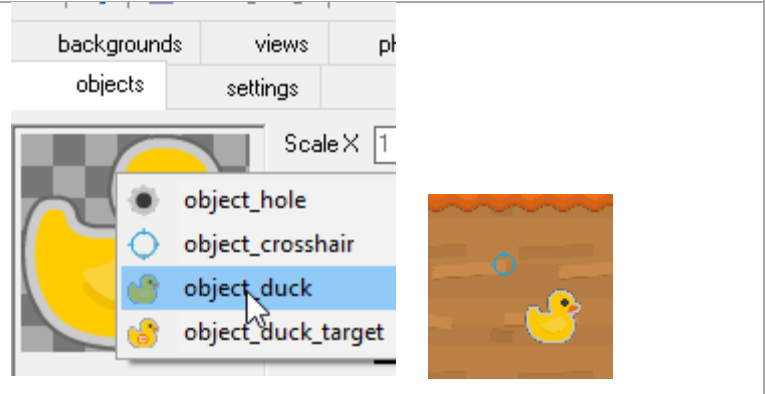
ANY TIME YOU WANT TO MODIFY AN EXISTING VALUE, USE RELATIVE. ANY TIME YOU WANT TO REPLACE AN EXISTING VALUE, DO NOT USE RELATIVE.



Test your code by adding a few ducks to the room and running the game.

They should disappear on their own but should disappear instantly if you click on them.

If you need to for testing purposes, change the timer in the Create event to 600 steps (10 seconds) so you have longer to shoot them.



Now right click the object_duck and **Duplicate** it
This will make a new object that has all the same events and actions.

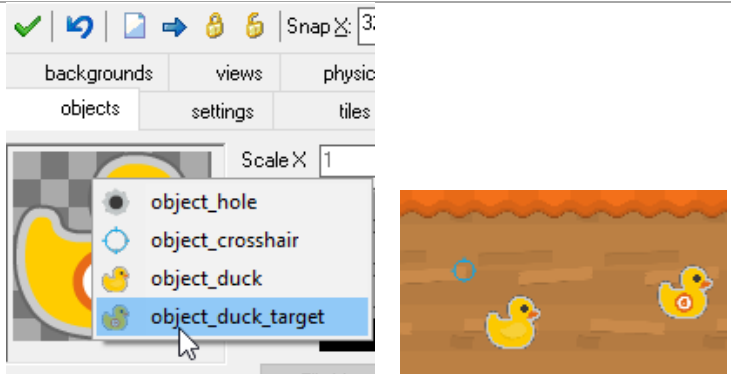
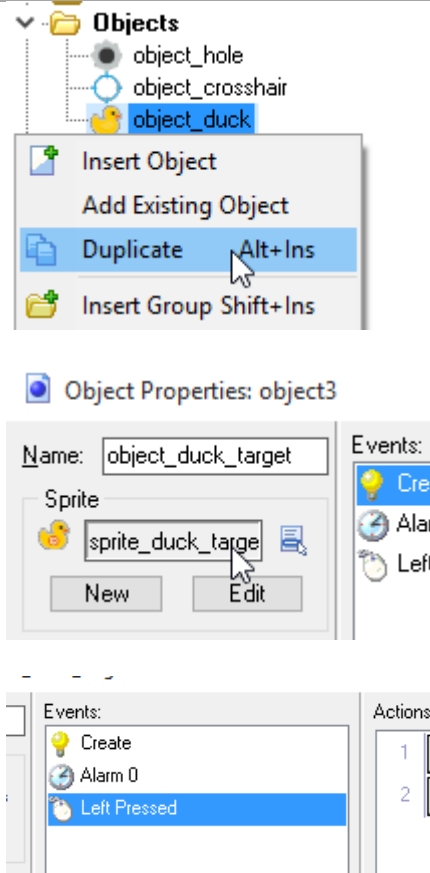
Change the name to **object_duck_target** and the
sprite to the target duck sprite.

Double click the **Set Score** in the **left pressed** and
change it to give the player 20 points. To hit this duck
is more challenging, they must click on the small
circular collision mask we set up in the sprite.

Test the target duck by placing a few in the room.

Make sure you can't take them out by clicking on
their heads.

Once again, temporarily. change their alarm timer if
you need more time to click on them.

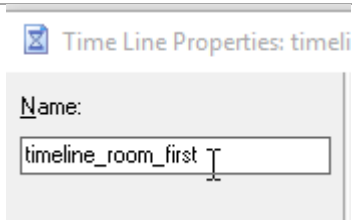


9 TIMELINE & CONTROLLER

Under **Resources** menu, create a **Timeline**

Call it **timeline_room_first**

A timeline is the easiest way to set up a sequence of
events – it will control our targets appearing.



Click **Add** and then type 75

This will make a timeline event after 75 steps (2.5 seconds).

Drag a **Create Instance** in as the action to take at step 60

Select the duck object and set the y to be **595** (right near the top of the waves) and the x to be **random(1024)**

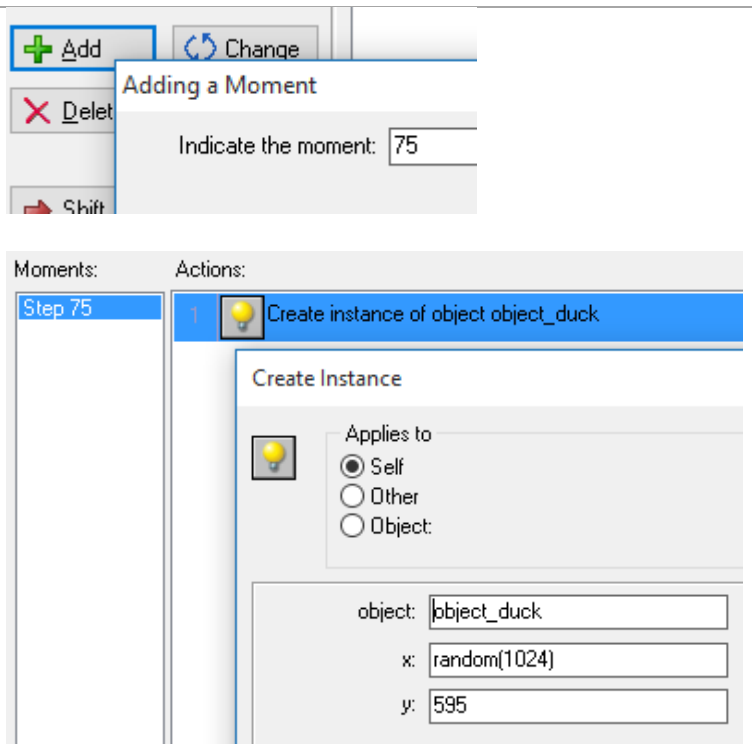
Make sure to use parentheses around the 1024 (not square braces).

random(1024) means pick a random number from 0 to 1023. Since our room is 1024 wide, the possible x locations are 0 to 1023.

This does mean the duck can appear halfway off the screen. If that bothers you, change the x to **random(824) + 100**

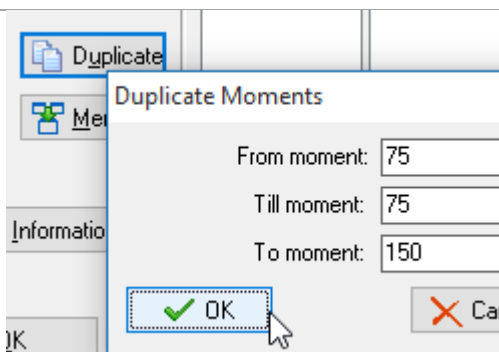
That means “pick a number from 0 to 823 and add 100” which gives between 100-923. That will mean the center of the duck is always at least 100 away from the edge of the screen.

These coordinates are NOT relative – they are an absolute location on the screen.



Duplicate the Step 75 to Step 150

This will make an exact copy that happens at 5 seconds (150 steps)




Add a moment for 225

Create a target duck at this time point


Use the same values for x and y
(Image shows the x that will not let the duck touch the edge).

Moments: Actions:

Step 75
Step 150
Step 225


1  Create instance of object <undefined>

Create Instance

 Applies to
☒ Self
☐ Other
☐ Object:

object:
x:
y:

☐ Relative

 OK

Finally, add a step 300, where you **Set the timeline speed** (Find it under the main 2 tab of actions)

Set the speed to **0.1 Relative**

Normally the timeline runs at a speed of 1


Changing the speed relative means when we repeat it, it will be running at 1.1 (10% faster).

Once it repeats twice and we get back to this step it will then change by 0.1 again to 1.2 (20% faster)


This will keep making the objects appear faster and faster as the game goes on, giving the player less time to react between them.

Moments: Actions:

Step 75
Step 150
Step 225
Step 300


1  Set time line speed to 0.1

Time Line Speed

 Applies to
☒ Self
☐ Other
☐ Object:

speed:

☒ Relative

 OK


Create **object_controller**

It will have no sprite.


This will be an invisible object that creates the targets, displays the score and otherwise controls things.

Object Properties: object4

Name:


Sprite
 

☒ Visible ☐ Solid
☐ Persistent ☐ Uses Physics

Add a **Create** event and place a **Set Time Line**  action in it.

Pick the timeline you just made and make sure to set it to Loop so it runs forever.


object_controller


no sprite> 

ew

le ☐ Solid
stent ☐ Uses Physics

0

<no parent> 


<same as sprite> 

<None>

Events:


Create

Actions:


1  Set time line <undefined>


Set Time Line

Applies to
☒ Self
☐ Other
☐ Object:

time line: 

position:

start: 

loop: 

Add E


Then add a **set score** to 0 – just to make sure that when the game starts, our score is set to 0


(It should be anyways... but in general the controller is a good place to make sure values are set appropriately – it will not get destroyed and recreated a lot – the player object might)

Events:


Create

Actions:

1  Set time line timeline_room_first

2  Set the score to 0

Set Score

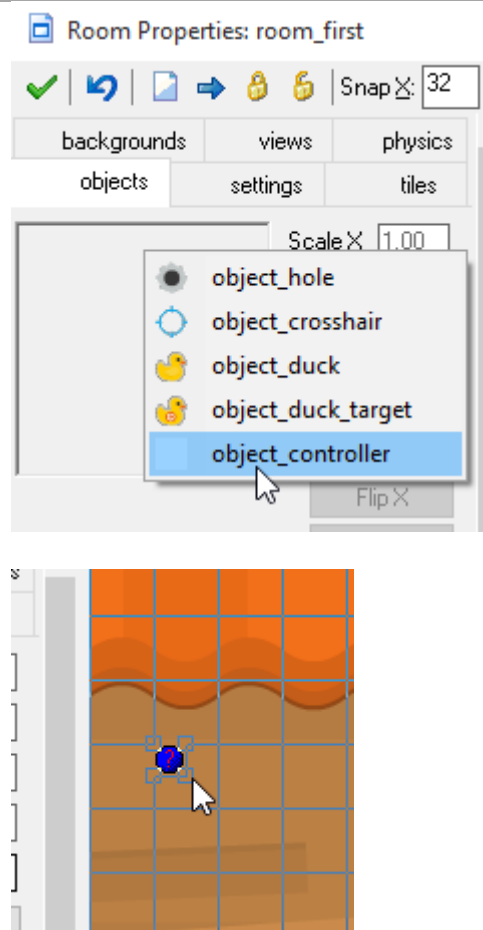


new score:

Go to the room and add a controller object – it will just show up as a blue circle with a red question mark since it does not have a sprite.

You only see this ? symbol in the room editor. When the game is running we will not see it at all.

The only things in your room should be the controller and crosshairs.



Test your game – ducks should appear randomly getting faster and faster.

10 FONTS & SCORE

To draw text in a game you have to create a font.

Under **Resources** menu, create a **Font** called **font_in_game**

You can make it whatever you like – I chose Bold Comic Sans

24 points is a good size

Add another **font_end** that is a bit bigger – like 36 points

The image displays two screenshots of a 'Font Properties' dialog box, likely from a game engine like Unity. Both screenshots show the 'font_in_game' and 'font_end' fonts, which are set to 'Comic Sans MS' and 'Bold'.

Font Properties: font_in_game

- Name: font_in_game
- Font: Comic Sans MS
- Style: Anti-Aliasing: 3, Size: 24
- High Quality: ☒ Bold: ☒ Italic: ☐

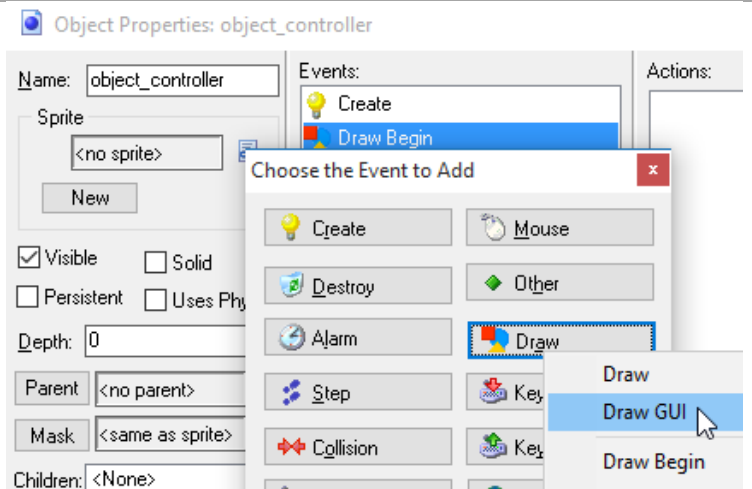
32 to 127

Font Properties: font_end

- Name: font_end
- Font: Comic Sans MS
- Style: Anti-Aliasing: 3, Size: 36
- High Quality: ☒ Bold: ☒ Italic: ☐

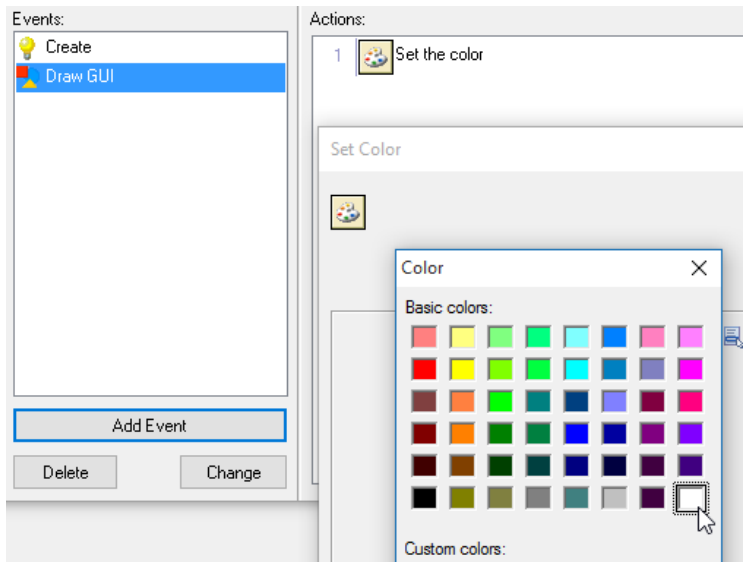
32 to 127

Go back to the controller object and add a **Draw GUI** event



Add a **Set Color** action from the **draw** tab of actions

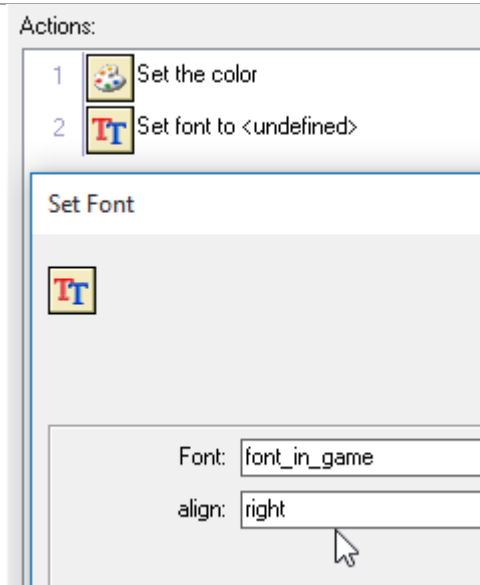
Select white as the color – note that you have to click the color bar and select white. If you leave the default “white” it will not actually use white.



Then add a **Set Font**

Choose your in game font and set it to align right

Align right means that the right side of the text will be lined up with the location we specify.







Finally, add a **Draw Score** from the scores tab of actions

Set it to draw a little away from the right edge of the top of the screen: x 1014 and y 0
These are NOT relative coordinates – they are absolute screen coordinates.

ANY DRAW _____ ACTION ONLY WORKS IF IT IS IN A DRAW OR A GRAW GUI EVENT. IF YOU PUT SOMETHING LIKE DRAW SCORE IN A CREATE EVENT OR SOMEWHERE ELSE, IT WILL DO NOTHING!

Actions:

- 1  Set the color
- 2  Set font to font_in_game
- 3  Draw a text
- 4  Draw the value of score

Draw Score



x: 1014

y: 0

caption: Score:

☐ Relative

Test your game – make sure you see your score and that it goes up as you shoot ducks.

11AMMO

Gamemaker has three built in variables – score, lives and health. To keep track of the player's ammo we will use lives. (Each bullet is kind of like a life)

Go to the **object_controller Create** and add a **Set Lives** action (score tab)

Set them to 8

Do NOT use relative – we want to set the value to 8, not change it by 8.

object_controller

<no sprite>

new

le ☐ Solid

istent ☐ Uses Physics

0

<no parent>

<same as sprite>




: <None>

Events:

Create

Draw GUI

Actions:

- 1  Set time line timeline_ro
- 2  Set the score to 0
- 3  Set lives to 0

Set Lives



new lives: 8

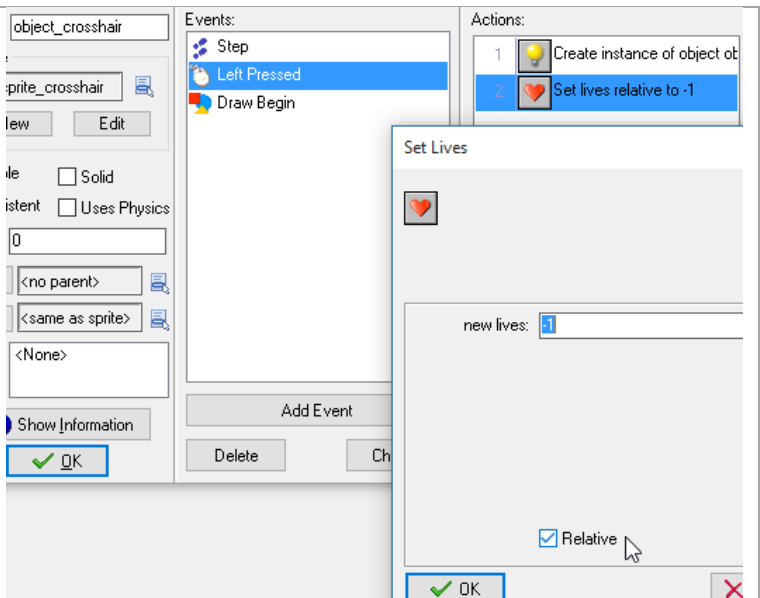
Now go to **object_crosshair** and add a **Set Lives** to the **Global Left Pressed** action.

Set the lives to **-1 relative**

That will subtract 1 from the current number of lives every time the player “shoots”

Note that score, lives and health are shared by all objects. That is why we can set them to 8 in the controller and then subtract from the lives here in the crosshair.

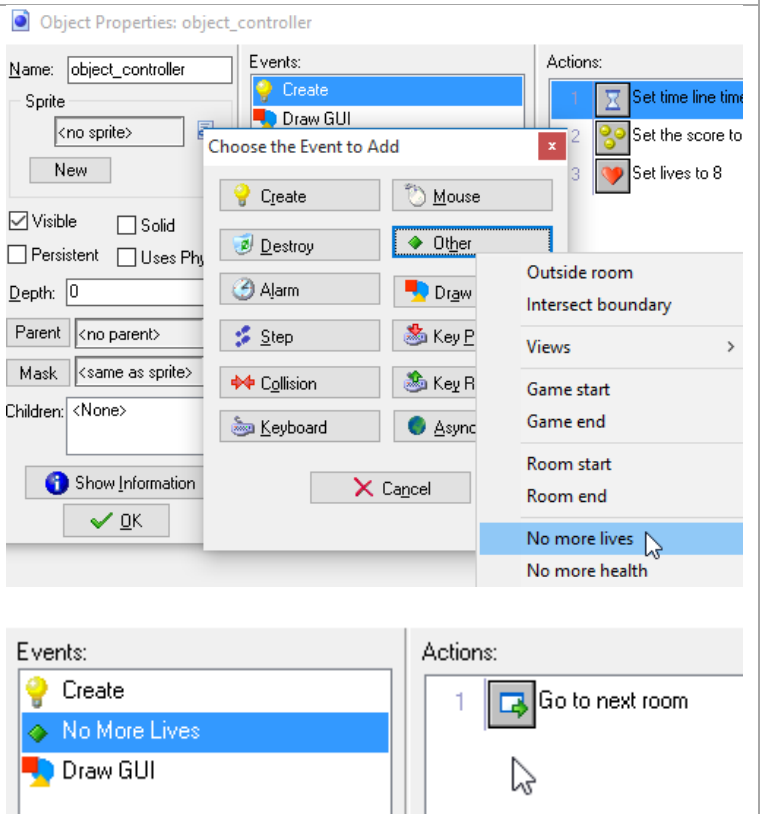
THIS ALSO MEANS YOU CAN'T USE LIVES (OR HEALTH OR SCORE) FOR TWO DIFFERENT PURPOSES – IF I MAKE A TWO PLAYER GAME AND TRY TO GIVE BOTH PLAYERS LIVES, THEY WILL ACTUALLY SHARE THE LIVES.



Go back to the controller

Add an **Other** event and select **No More Lives** event
This event fires when we run out of lives (ammo)

Note that because lives are shared by everything, we could put this action in the crosshairs or the controller... it would work the same either way. I am choosing to consolidate most of the “scoring” logic here in the controller.

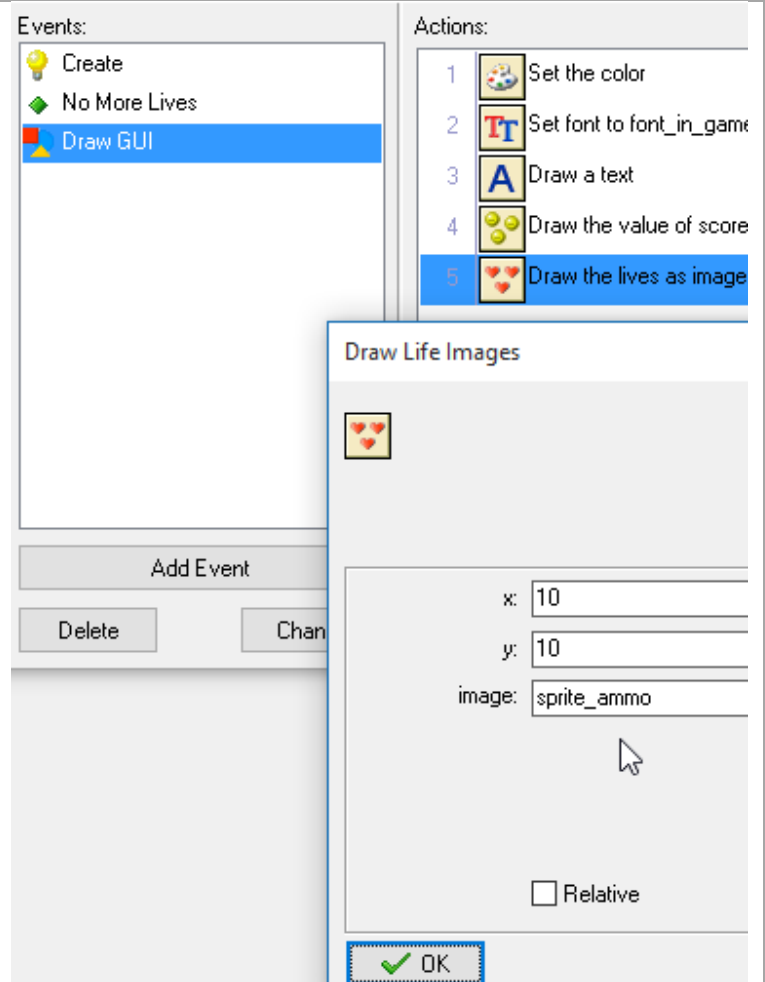


Add a **Go to Next Room** action (main 1 tab) to happen when we are out of lives.

Go to the **Draw GUI** of the controller and add a **Draw Lives as Image**

This will let us pick a picture that is draw as many times as we have lives left (ammo).

Choose the ammo sprite and set it to draw at 10, 10 (absolute coordinates). This is 10 pixels over and down from the upper left corner of the room.



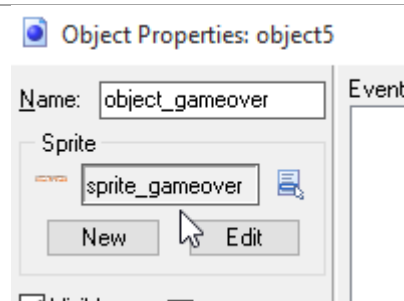
The screenshot shows the Scratch IDE's 'Events' and 'Actions' panels. In the 'Events' panel, 'Draw GUI' is selected. In the 'Actions' panel, 'Draw the lives as image' is selected. A 'Draw Life Images' dialog box is open, showing the 'ammo' sprite and the coordinates (10, 10) for the image. The 'image' field is set to 'sprite_ammo'.

Test the game.

You should see your ammo. Running out of ammo should take you to the next room which will crash the game (there is no next room!). We will fix that in a little bit.

12END GAME

Make **object_gameover** using the **appropriate** sprite



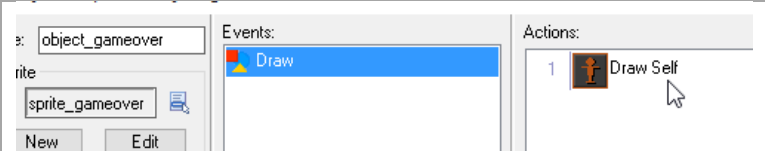
The screenshot shows the 'Object Properties: object5' panel. The 'Name' field is set to 'object_gameover'. The 'Sprite' field is set to 'sprite_gameover'. There are 'New' and 'Edit' buttons below the 'Sprite' field.

Add a **Draw** event

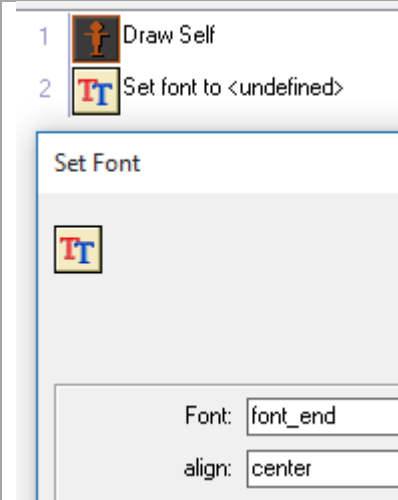
Draw event is used when you want to replace the normal drawing of an object (its sprite) with something fancier.

Add a **Draw Self** action to it. When you make a Draw event, the sprite will not be drawn unless you tell it to be – Draw Self says “draw my normal sprite”

(Draw GUI event would work fine for this logic also)



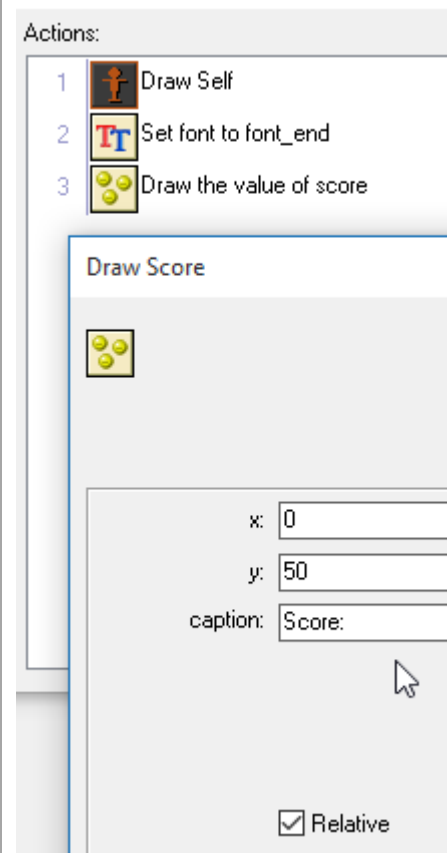
Set the font to **font_end** and center it.



Then **Draw Score** at 0, 50 Relative.

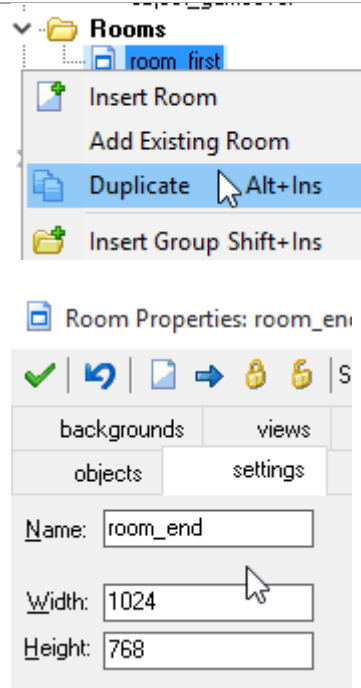
The sprite will be drawn wherever we place the object. The score will be drawn 50 pixels below that (0, 50 relative is over 0, down 50).

Since we centered the font and we centered the gameover sprite, they should be lined up perfectly.



Duplicate the first room – this way we get all the backgrounds and objects from the first room

Call it **room_end**

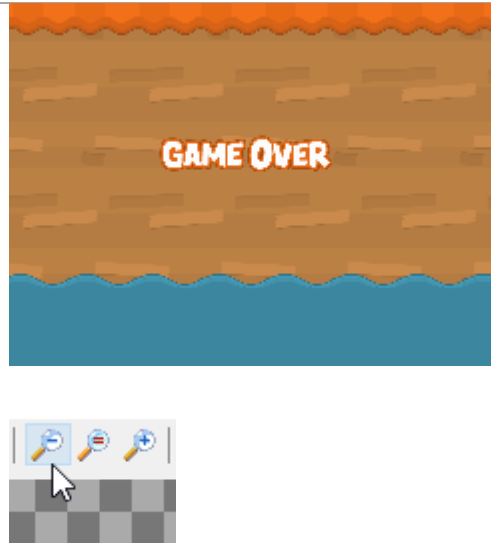


CTRL-Right Click the crosshair and controller to remove them from this new room – the player doesn't actually play in this room – just sees their score.

Place a **object_gameover** in the middle of the room.

Tips:

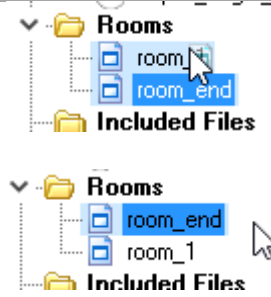
- You can Zoom the room view using the magnifiers at the top of the window
- Holding space and click dragging will move the room around in the window



Test the game over room. Click and drag room_end up before room_1.

This will make the game start in the end room.
Rooms are ordered in game in the same order you place them in the rooms section of the resources window.

If it shows you your score (0), move it back to after room_1 and try playing the game. You should see your score after running out of ammo.



13 MORE TARGETS

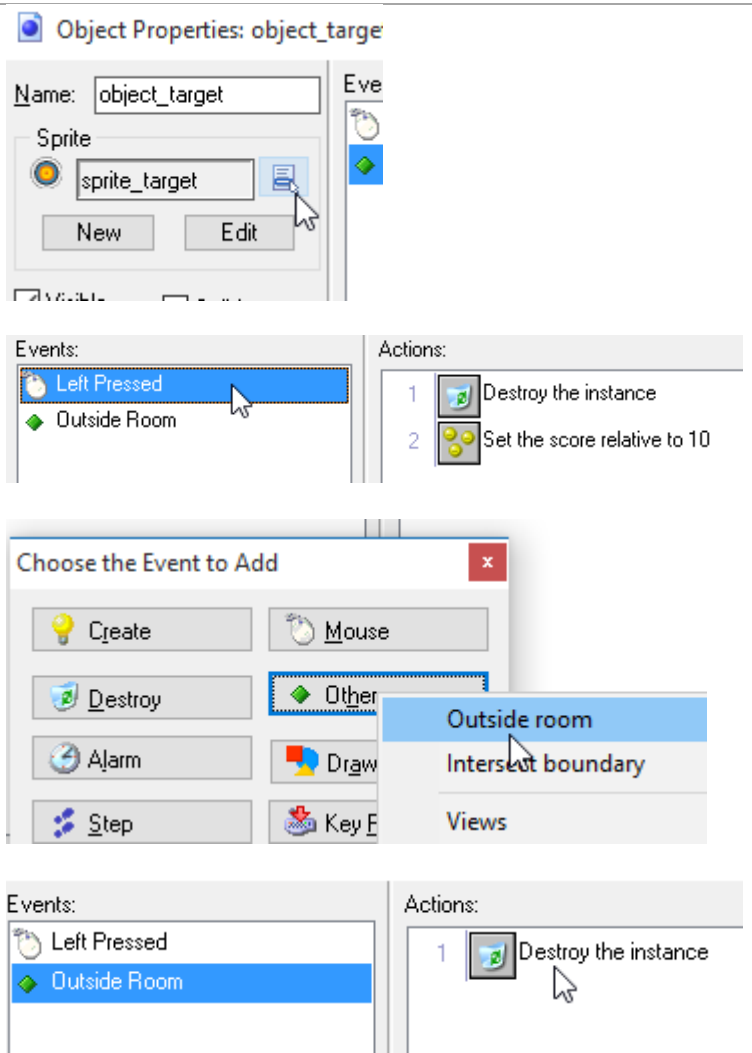
Add an **object_target** and use the matching sprite

Add a Left Pressed (not global) and destroy it and add 10 points. (Set score 10 relative).

Then add **Other, Outside Room**

This event fires when the object leaves the room. It is a good way to destroy things that have flown away so that gamemaker isn't keeping track of 1000's of old objects that we never will actually see again.

Destroy it when it leaves the room.



Make an **object_target_bullets**


Make it the same as the target object, but when left button is pressed on it, we want to add three lives (bullets) instead of giving points.





Set lives to 3 relative




When it leaves the room it is destroyed.

Object Properties: object_target_bullets

Name:

Sprite: 

Events:	Actions:
 Left Button	1  Destroy the instance
 Outside Room	2  Set lives relative to 3

Events:	Actions:
 Left Button	1  Destroy the instance
 Outside Room	

Go to the timeline

Add a **Step 90**

Add **Create Moving**

This makes an object and sets it in motion all at once.

Pick **object_target_bullets**

Set x to **960** (near the right side of the screen)

Set y to **random(300) + 100**

This will pick a number from 0 to 299 and add 100 to it. Thus y is set to something between 100 and 399.

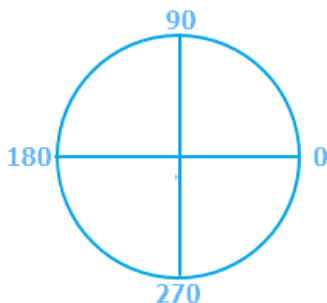
That is below the curtain but above the water.


Set the Speed to **12**

Set the Direction to **180** (left)


The bullet target appears on the right side and moves across to the left.

HERE IS HOW DIRECTIONS WORK:



Moments:	Actions:
Step 75 Step 90 Step 115 Step 150 Step 225 Step 270 Step 300	1  Create moving instance of object_target_bullets

Create Moving

 Applies to:
☒ Self
☐ Other
☐ Object:

object:

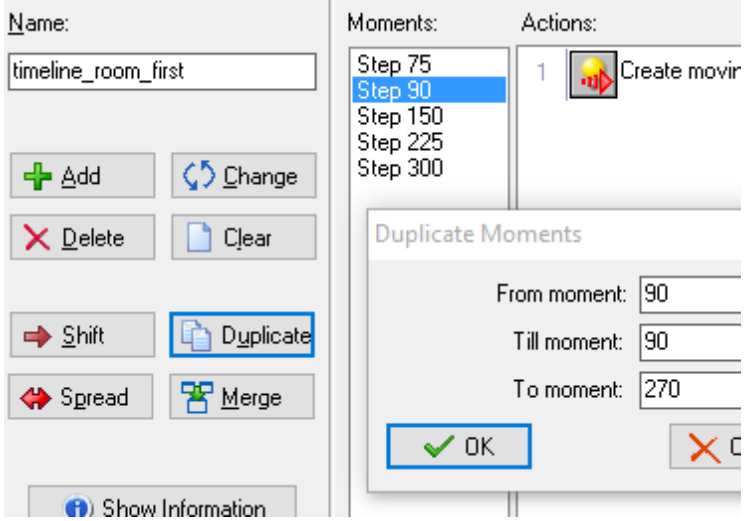
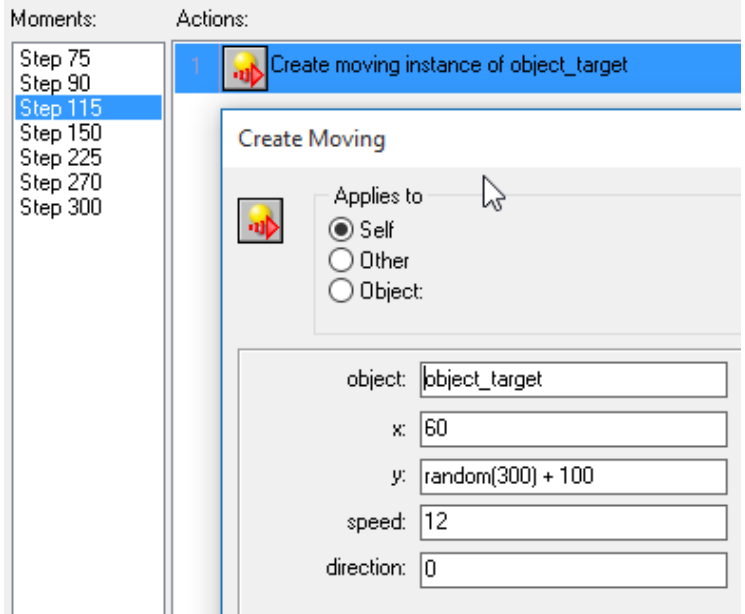
x:

y:

speed:

direction:

☐ Relative

<p>Duplicate step 90 to moment 270</p> <p>There will be two bullet targets to shoot per “round” of the timeline.</p>	
<p>Make a Step 115 on the timeline</p> <p>Create a moving regular target</p> <p>This one should start on the left side (x: 60 is near the left edge) and move to the right (direction 0). It moves a little slower than the bullet target.</p>	
<p>Try the game. You should see both kinds of target appear during the course of the game. Shooting the ammo target should give you 2 more bullets. (Shooting costs one, hitting the target gives you back 3).</p>	

14A TIMER

You may have noticed that you can just sit in the game forever lining up shots. If you are careful you will never run out of ammo!

A timer will add some urgency to the game and ensure it ends.

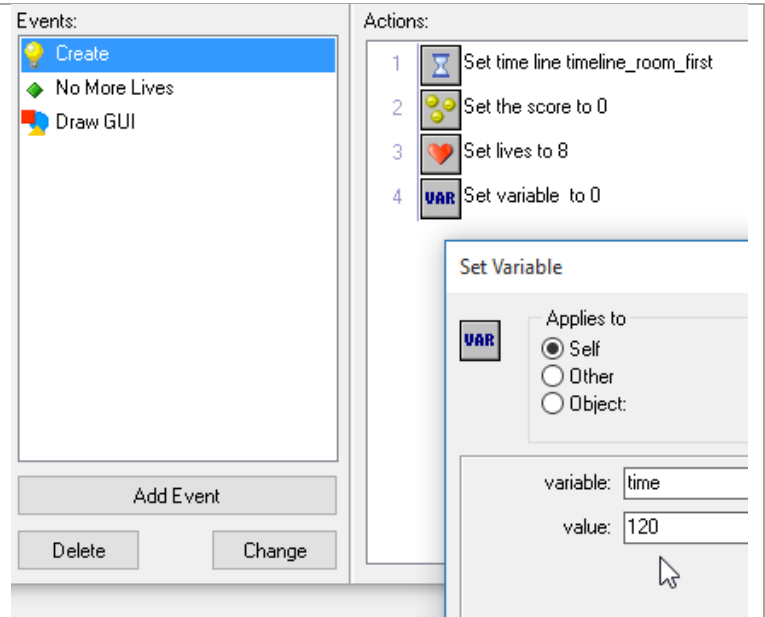
Go back to the controller object's **Create**

Add a **Set Variable** (from control tab of actions)

Set variable allows you to store a piece of information (usually a number).

We are using it here to store the number **120** and calling the stored value **time**

To us it represents the idea that there are 120 seconds left in the game. To the computer, it is just a piece of information called "time" that has the value 120. It could be called "bob" and it would be all the same to Gamemaker



Also in **Create**, set **Alarm 0** to **30** steps
Thus Alarm 0 will go off after 1 second (30 steps)

Add an **Alarm 0** event to handle when the alarm goes off.

To it add **Set Variable** again

This time, use **time** and **-1 Relative**

When the alarm goes off, we want to subtract one (-1 relative) from the piece of information called "time".

SPELLING AND CAPITALIZATION COUNT!!!

IF YOU BY ACCIDENT TYPE TIME DIFFERENT IN THE TWO SET VARIABLES (SAY TIME VS TIME OR TIME VS TIEM), GAMEMAKER WILL GET CONFUSED AND YOU WILL LIKELY GET AN ERROR LIKE THE ONE SHOWN BELOW. IN IT GAMEMAKER IS SAYING "I CAN'T GET THE VALUE OF TIEM THAT YOU ARE ASKING ME TO CHANGE... I DON'T KNOW WHAT THAT IS!"

```
#####  
FATAL ERROR in  
action number 1  
of Alarm Event for alarm 0  
for object object_controller:  
  
Push :: Execution Error - Variable Get 100001.tiem(100001, -214  
at gml_Object_object_controller_ObjAlarm0_1 (line 1) - tiem +=  
#####
```

Events:

- Create
- No More Lives
- Draw GUI

Actions:

- 1 Set time line timeline_room_first
- 2 Set the score to 0
- 3 Set lives to 8
- 4 VAR Set variable time to 120
- 5 Set Alarm 0 to 30

Events:

- Create
- Alarm 0
- No More Lives
- Draw GUI

Add Event

Delete

Change

Actions:

- 1 VAR Set variable to 0

Set Variable

Applies to
☒ Self
☐ Other
☐ Object:

variable: time

value: -1

☒ Relative

Now we need to check and see if we are out of time.

Add a **Test Variable** (again from control tab)
Type **time** and **0** then make sure that **equal to** is selected

The octagon shape means a conditional check. We are checking to see if a condition is true – in this case if the value “time” is equal to 0. If it is, we do the next action, if not, we skip it.

Put a **Go To Next Room** (from main 1) after the test
If time is 0, we go to the next room, if it is not 0, we do skip over that action.

Although the previous version would work fine, I recommend you always use the **start block** and **end block** actions (from control tab) after a conditional.

They make a group of actions that all depend on the test that comes before the **start block** triangle. We could add more actions between the two triangle and all would depend on the “is time = 0” check.

Also, they make the block of actions indent in the editor, which makes it easier to see what depends on the test.

Events:

- Create
- Alarm 0
- No More Lives
- Draw GUI

Add Event

Actions:

- 1 VAR Set variable time to -1
- 2 VAR If time is equal to 0

Test Variable

Applies to

- ☒ Self
- ☐ Other
- ☐ Object

variable: time

value: 0

operation: equal to

Actions:

- 1 VAR Set variable time to -1
- 2 VAR If time is equal to 0
- 3 Go to next room

Actions:

- 1 VAR Set variable time to -1
- 2 VAR If time is equal to 0
- 3 Start of a block
- 4 Go to next room
- 5 End of a block

Question

Other

Finally we need to reset the alarm clock so it goes off again.

To **Alarm 0** event add a **Set Alarm 0 to 30** action

This may sound odd to set the alarm again in the alarm event, but it is a good way to “hit the snooze button” and make sure we do the same thing in another second (30 steps). When the alarm goes off then, we will subtract another 1 from time, see if we are at 0, and reset the alarm yet again.

Note that ONLY the **Go to next room** depends on the **Test Variable time = 0** We are always going to do the **Set Alarm 0 to 30**

The octagon shaped conditional only affects the next block in the bottom version. Notice that the top version makes it MUCH clearer that the Set Alarm does not depend on the If time = 0. That is why I encourage you to put the start and end blocks in even if they are not technically needed here.

Events:

- Create
- Alarm 0
- No More Lives
- Draw GUI

Actions:

- 1 **VAR** Set variable tiem to -1
- 2 **VAR** If time is equal to 0
- 3 Start of a block
- 4 Go to next room
- 5 End of a block
- 6 Set Alarm 0 to 30

OR

Events:

- Create
- Alarm 0
- No More Lives
- Draw GUI

Actions:

- 1 **VAR** Set variable tiem to -1
- 2 **VAR** If time is equal to 0
- 3 Go to next room
- 4 Set Alarm 0 to 30

Now we need to draw the time left – we will do so between the ammo and the score.

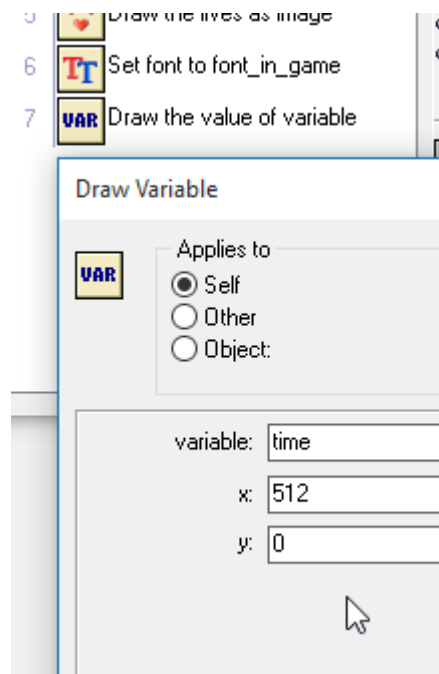
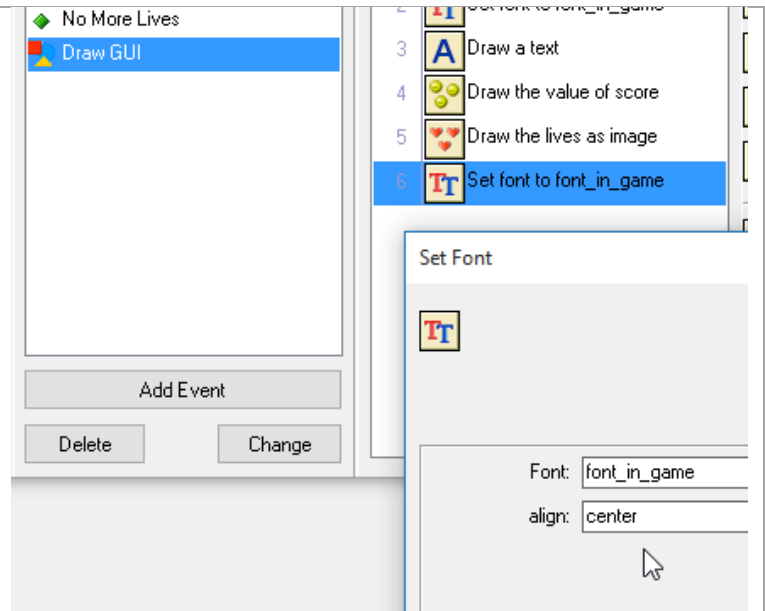
Go to the Draw GUI event of the controller

Add another **Set Font** at the end. Select the **font_in_game** font and align it to the **center**. (So when we place the text in the center of the screen it expands in both directions equally)

Finally, add a **Draw Variable** (from control tab) Specify to draw the value of **time** at 512, 0 (center of the top of the screen). These coordinates are absolute, NOT relative.

Gamemaker will look for the piece of information called “time” and then draw its value. We will not see the word time, just its value (initially 120).

If you wanted to see something like “Time: 120” You would have to first use a Draw Text event to draw the “Time: “ part.



Test your game.

To test that it ends when the timer runs out, you might want to temporarily change the Set Variable time to 120 in the create event to Set Variable time to 10. It will be a very short game but make it easier to test the timer if something is wrong.

15EXTRAS

Time to improve on the game.

Before you start making changes, make a backup copy of your project. Copy the WHOLE ShootingGame.gmx folder, or whatever you called it, to another location. You can NOT just copy the ShootingGallery.project.gmx file – it does not work on its own. Any time you try to back up a project or (submit one to me), you need to move the entire folder. If you start making changes and realize that they aren't working (which WILL happen to you at some point), you can easily go back to your saved copy of the project. Back up early and back up often.

You should spend a couple of hours experimenting and trying to add some features to the game. Depending on your experience working with code and/or art, you may get more or less far. You certainly should not know how to do everything at this point... there will likely be many ideas you come up with that you don't know how to implement. You can always ask for advice on the discussion boards or in office hours, but some things you may not be able to pull off yet. When you get an idea, give it a shot, if you get stuck, get help or move on to something else.

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.

IDEAS:

Note that many of these ideas are “more of the same”... they don't add a whole lot in the way of new game mechanics. In general, those are probably the least interesting things to add to a game. But, they are also the easiest to figure out. Since you are likely just getting started with Gamemaker, you likely want to start off with an easy thing or two before trying something more creative.

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.

- Modify the timeline – make targets appear at different locations or that go from the top to the bottom, etc...
- Make a new kind of target – maybe it moves faster or is smaller or larger or rarer. Try to balance the points with the difficulty. Maybe it is a “bad” target you lose points for shooting.
- Make a room_2 that uses the grass background and different kinds of motion (all targets fly up from the ground into the sky?). Make a new timeline for room 2. Duplicate the object_controller to make object_controller2 that you place in room_2. It should not set the **lives** or **score** in its create event (so you keep the ones from room_1) and should start the new timeline instead of the room_first timeline.
- Read the manual section on paths (<http://docs.yoyogames.com/index.html?page=source%2Fdadiospice%2Findex.html>) and make a target that moves on a curvy path across the screen.