Computer Science Department

COMPX251: GAMES THEME
Making Interactive Computer Games

The objective of this theme is to explore some of the issues involved in programming computer games. The software used is Game Maker Studio 2. It provides an interactive environment that allows us to construct a variety of 2 dimensional games (platform, maze, scrolling, etc) without having to be aware of detailed implementation issues.

DOCUMENT LIST

Additional help is available from the Game Maker online help system and from the Game Maker web site. The web site has links to examples, tutorials and so forth. Also, if you would like to try the software at home, you can download the free trial version from the web site. The web site address is http://www.yoyogames.com/get. The ‘free’ version of the software is available free of charge, and is suitable for the exercises in this module. .

GAME MAKER INTRODUCTION

From Mark Overmars, the original creator of Game Maker

Before delving into the possibilities of Game Maker it is good to first get a feeling for the global idea behind the program. Games created with Game Maker take place in one or more rooms. (Rooms are flat, not 3D, but they can contain 3D-looking graphics.) In these rooms you place objects, which you can define in the program. Typical objects are the walls, moving balls, the main character, monsters, etc. Some objects, like walls, just sit there and don't do anything. Other objects, like the main character, will move around and react to input from the player (keyboard, mouse, and joystick) and to each other. For example, when the main character meets a monster he might die. Objects are the most important ingredients of games made with Game Maker, so let us talk a bit more about them.

First of all, most objects need some image to make them visible on the screen. Such images are call sprites. A sprite is often not a single image but a set of images that are shown one after the other to create an animation. In this way it looks like the character walks, a ball rotates, a spaceship explodes, etc. During the game the sprite for a particular object can change. (So the character can look different when it walks to the left or to the right.) You can create your own sprite in Game Maker or load them from files (e.g. animated GIF's).

Certain things will happen to objects. Such happenings are called events. Objects can take certain actions when events happen. There are a large number of different events that can take place and a large number of different actions that you can let your objects take. For example, there is a creation event when the object gets created. (To be more precise, when an instance of an object gets created; there can be multiple instances of the same object.) For example, when a ball object gets created you can give it some motion action such that it starts moving. When two objects meet you get a collision event. In such a case you can make the ball stop or reverse direction. You can also play a sound effect. To this end Game Maker lets you define sounds. When the player presses a key on the keyboard there is a keyboard event, and the object can take an appropriate action, like moving in the direction indicated. I hope you get the idea. For each object you design you can indicate actions for various events, in this way defining the behaviour of the object.

Once you have defined your objects it is time to define the rooms in which they will live. Rooms can be used for levels in your game or to check out different places. There are actions to move from one room to another. Rooms first of all have a background. This can be a simple colour or an image. Such background images can be created in Game Maker or you can load them from files. (The background can do a lot of things but for the time being, just consider it as something that makes the rooms look nice.) Next you can place the objects in the room. You can place multiple instances of the same object in a room. So, for example, you need to define just one wall object and can use it at many places. Also you can have multiple instances of the same monster objects, as long as they have the same behaviour.

Now you are ready to run the game. The first room will be shown and objects will come to life because of the actions in their creation events. They will start reacting to each other due to actions in collision events and they can react to the player using the actions in their keyboard or mouse events.

So in summary, the following things (often called resources) play a crucial role:

- objects: which are the true entities in the game
- rooms: the places (levels) in which the objects live
- sprites: (animated) images that are used to represent the objects
- sounds: these can be used in games, either as background music or as effects
- backgrounds: the images used as background for the rooms

There are actually a number of other types of resources: paths, scripts, data files, and time lines. These are important for more complicated games. You will only see them when you run Game Maker in advanced mode. They will be treated in the advanced chapters of the documentation.

GAMES THEME – Practical Module One (P1A)

Clicky to Space Invaders

Sessions:

1. Getting Started
2. Events
3. Controlling a piece
4. Bullets
5. Parenting

Games – Clicky to Space Invaders – Session One

Getting Started: Introduction to Game Maker

This session starts the 'Clicky' game. It gets you to the point of having an object moving on the screen, but not yet to having anything interactive running. The main objective is to familiarise you with the layout and operation of the Game Maker user interface.

A number of files are provided with this module (images and sounds). Make a copy of the files for this module (from directories GamesClicky and GamesInvaders). The files can be found in the COMPX251 folder in the L: library

---

When asked to load files, you can choose your own from the collection of files that are part of the Game Maker installation, or you may chose the ones provided. The ones provided will include images that are of the correct size, which is sometimes important (see instructions)

At the end of this session, you should be able to…

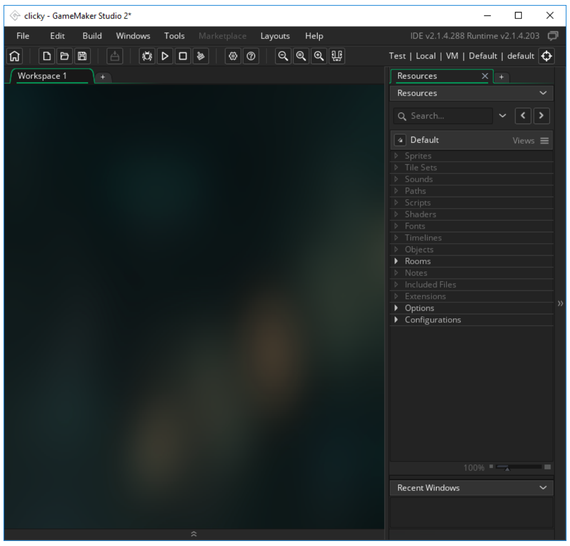
- Create a room/level and decorate it with a background.
- Create or load your own sprites and using those, create objects.
- Create a basic event so that your objects can start to do something.

To start Gamemaker Sudio 2, go to the Start menu and find Game Maker Studio 2, click on the sub folder and choose Game Maker Studio 2. It will then ask you to log into an account. The username is already filled in for you but you will need to type in the password **quidditch** and it will log you into a university account created for Game Maker Studio 2 in the lab. It may take a few minutes to start up.

If you wish to use Game Maker Studio 2 at home you can use the username and password assigned to you.

The start screen will appear, click on the New button and it will ask you to choose a template. Click on the New Blank button and then choose Drag and Drop to create a Drag and Drop project. The Save As dialog window will appear so navigate to your H: drive and into the folder you wish to create your game in and then type in a project name and click on the **Save** button. Game Maker will create a new project folder for you in the folder that you selected.

You should end up with a window similar to that in the image below.



The default skin (theme) for Game Maker studio is the dark skin displayed above. There is a light skin also available. If you wish to try it go to the File menu and select Preferences and select General Settings. Then change the IDE Skin to light and click on apply, you will need to restart Game Maker to see the changes and then select the Open button and find your project in your account and open the project by selecting the **.yyp** file. Feel free to use whichever skin you prefer, we will use the dark skin for the screenshots in this module.

This is the editor, where you will build and edit your game. On the right side of the window are the folders ('Sprites', 'Sounds', …) which will hold game assets you will load or create later. Along the top is first the menu bar (File, Edit, …) and below that the toolbar with 6 groups of buttons. The first (leftmost) group is : 'Create a new game', 'Open an existing game', 'Save the game'. These two icons, , are the Play and Stop buttons to allow you to start the game and stop the game. Move the mouse pointer over the other tool bar buttons to see the pop ups about what that button is for.

The first part of this module is an introduction to the operation of Game Maker. You will create an object inside an enclosed area, and make it bounce off the walls. Then you will make it into a test of skill, where you have to click on an object as it moves. Your tasks are:

1. Create a background.
2. Create a sprite.
3. Create a room.
4. Decorate your room with the background.
5. Create objects, and set up their basic properties.
6. Place these objects in your room.
7. Make the objects interact with you (the player).

<u>Stage 1. Creating the background sprite.</u>

A 'room' is Game Maker's name for a level in a game – a place where the game takes place. The room used in our first game will have a 'background' – a picture that fills the game window. We will later place objects on top of the background – for example there will be wall objects around the side to stop items from flying out of the game. The background is required so that objects will draw properly, but the image chosen is up to you. You can use a simple coloured background, or something more interesting. A background image is a sprite (graphic asset) which is placed in the background of a room.

In the **Resources** window on the right hand side, right click on the **Sprites** folder and then select **Create** and then select **Sprite** which allow you to load an image file for the sprite. Click on the **Import** button and then then navigate to the **GamesClicky** folder and select the **Stary Background** file. The Sprite window will now appear with a preview of the image. Change the name from **Sprite1** to **StaryBackground** and then close the window. When coding you always give good names to assets so that it is easier to use those assets.

Stage 2. Creating more sprites.

Now, you need to make the object that will be bouncing around in our room. For this, you need two things. First of all, you need a sprite. A sprite is a 2 dimensional image which you will be using to represent the main object. And the second thing you need is the object itself.

- Create a new sprite just like you did for the background sprite and name it **sprButton**. Use the button.bmp file for the image.

You will notice that there is a white background around the button image, this would look horrible against the black starry background. What we want to do is to make the area outside the button transparent so that even though the sprite is a square (32 pixels wide by 32 pixels high) we only want to see the button area of the sprite.

Click on the **Edit Image** button to bring up the sprite editor in a new tab. Then select the **Color Remove** tool (), any colour you click on in your sprite image will now be removed. Click on a white area outside the button image and all the white colour in the sprite will be removed. Close the Sprite Editor tab to go back to the Sprite window and then close the sprite window.

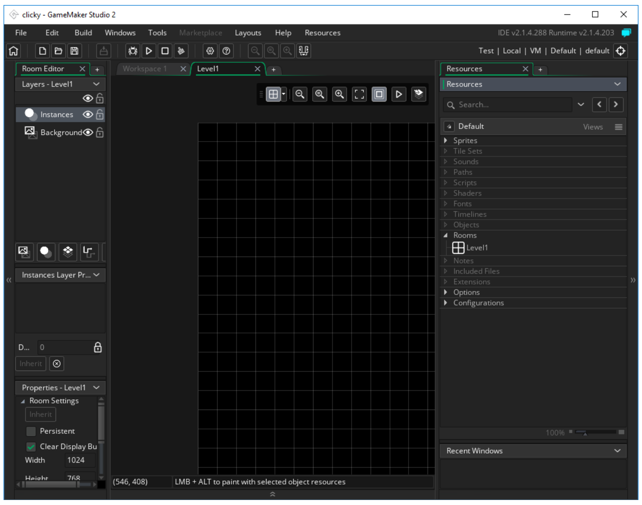
You will now have a background sprite and a sprite for your object, but you also need a wall sprite as the image for making walls to contain your object.

- Add another sprite, as you just did before, but choose 'Wall.gif' and name it 'sprWall'.

Now, you need to make a new room/level.

Stage 3. Setting up your first room/level.

A room has already been created called Room1. Double click on the Rooms folder in the Resources window to see the room. Right click on **Room1** and select **Rename**, change the name of the room to **Level1** and then double click the room to open it in a new tab. You should get something like this…



Persistent
Clear Display Bu
Width 1024
Height 768

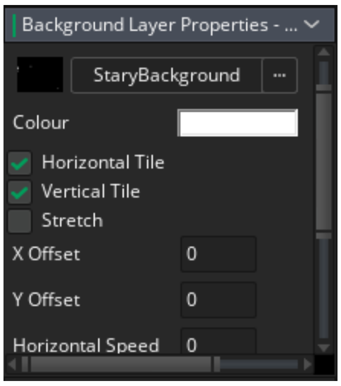
(546, 408) LMB + ALT to paint with selected object resources

This is your room construction page. In here, you'll place your background sprite and objects so that they can interact with each other. At the moment, you'll see that there's nothing but a black background in the room, there is also a grid shown. Click on the down arrow next to the Grid button to display the Grid Options, and change the Grid Colour to a colour you prefer.
On the left hand side you will see the Properties window for Level1. Change the width to **640** and the height to **480**. This is how you can set the dimensions for your game levels.

On the left hand side in the Layers window select the **Background** option, and you will see the Background Layer properties window appear. Click on the dropdown button that currently says **no Sprite** and then choose the StaryBackground sprite. You will see the background sprite appear but it won't fill the whole area of the level. Tick the Horizontal Tile and Vertical Tile options to see what they do. Leave them ticked so that the background fills the entire level area.

Press the play button

<u>Stage 4. Placing a background in your room.</u>


Background Layer Properties - ...

StaryBackground

Colour

✓ Horizontal Tile
✓ Vertical Tile
Stretch

X Offset	0
Y Offset	0
Horizontal Speed	0

On the left hand side in the Layers window select the **Background** option, and you will see the Background Layer properties window appear. Click on the dropdown button that currently says **no Sprite** and then choose the StaryBackground sprite. You will see the background sprite appear but it won't fill the whole area of the level. Tick the Horizontal Tile and Vertical Tile options to see what they do. Leave them ticked so that the background fills the entire level area.

.

Close the Level1 tab and then press the 'Run' button in the toolbar, or open the Build menu, and select Run. You should see, after a short pause, a window that shows up with your room and your background, but nothing else.

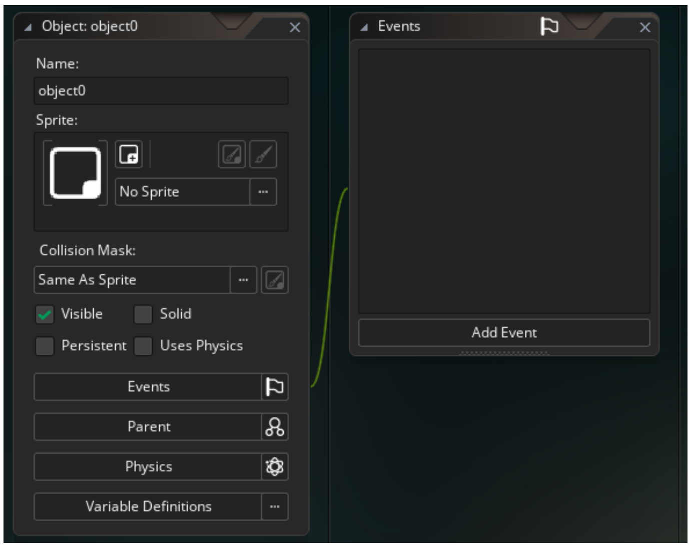
You may see a windows message pop up about opening the Game Bar, if you move your mouse over this message you can tick the don't show again option.

Close this window to return to the Game Maker program. You should save your game now and do so quite often.

Now, to further construct the room. You need to give it boundaries… Walls that an object will bounce off. For that, you need to define an object with our wall sprite as its image.

<u>Stage 5. Create objects, and set up their basic properties.</u>

Right click on the **Objects** folder on the right hand side and select **Create Object**. You will see the Object editor window appear. You can scroll the window by clicking and holding the scroll wheel (middle mouse button) and dragging. Try it to get the hand of it.



Now, this is the defining part of this program. In here you create objects, give them actions, and let them interact when playing. The parts of this window are…

Name: The name of the object in the editor.

Sprite: An object does not come with a default sprite. This is where you can choose what image the object will have. This image will also be used in its collision detection.

Solid: There are two types of collision events that can be defined. Collision with solids, and collision with all. Here, you can set an object to become solid.

Visible: This tells the program whether or not the object is visible at the start of the game or not. Sometimes objects start invisible and are made to appear at some suitable point in play.

Persistent: Tells the game not to destroy that object when a room change occurs.

Uses Physics: Not used in this theme.

Parent: Discussed in later modules.

Variable Definitions: Discussed in later modules.

The window at the right of the Object Property window define possible actions (events) for objects – they may move for example. This module will only be using a couple, but we will introduce others at a steady pace. The full set is described in the help system.

Ok, now that the Object Property window is visible, a wall object can be created quite simply. Change the object's name to 'objWall' (it's better than relying on the image to distinguish it, especially in later games). Click on the sprite dropdown button which is currently set to No Sprite and click on your wall sprite.

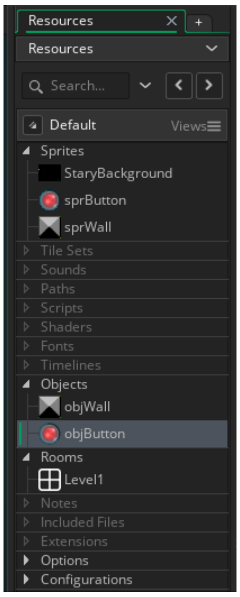
- Tick the Solid box to say that this object is solid (that means things can bounce off it).
- None of the other settings need to be changed, and there are no events that need to be created for this object, so press close the object editor window.

Now that you have your wall, you need another object. This is the object that will be bouncing around in your 'room'.

Repeat the previous steps. Create an object called 'objButton', give it its sprite, but **don't** make it solid.

If at any time you have pressed Ok, but still want to change an object (or any other asset), just double click on the object(asset)'s name in the objects folder to reopen the property window.

When you have your objects created, you should have something looking like this.



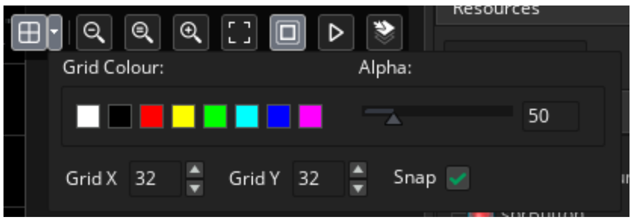
Note: If you want to change the name of your sprites at any time, you can do so without having to change the objects as well.

Now, you have the makings of a game. The last things that need to be done are finishing off creating the room, placing the objects that you have created, and then giving your objects their events and actions, which give them life.

Stage 6. Placing your objects in a room.

- Open your room's property window by double clicking on its name on the right hand side of the window, if it's not already open.

First thing you need to do is check the grid size. Since our walls and our object are both 32x32 pixels (at least they should be) then if the grid is 16x16 or other values, there will be a bit of trouble placing the objects. Click on the Grid Options button in the tool bar and make sure that both GridX and GridY are set to 32.



50

Grid X 32 Grid Y 32 Snap ✓

In the Layers window on the left hand side make sure Instances has been selected. This layer will display instances of your objects that have been added to your room. You must select the object that you want to add to the room by selecting it in the Resources window first. Select the button object and then with your mouse point inside the room area press the Alt key on your keyboard. You will see the object that will be added appear, release the Alt key and it will disappear. Select the Wall object and try pressing the Alt key again.

> To place an object in your room, simply left click while pressing the Aly key in the room where you want the object. It will snap to the top left hand corner of the grid square you clicked on. If you hold down the <Alt> key and click and drag, a line of objects will be placed. Then let go of the Alt key and click somewhere in the room to place those objects.

If you made a mistake at any time in placing your objects, simply left click an object instance to select it and press the Delete key on your keyboard. You can hold down the shift key while dragging to select multiple objects at the same time and press the delete key to remove all of them. Experiment with placing multiple object and deleting multiple objects.

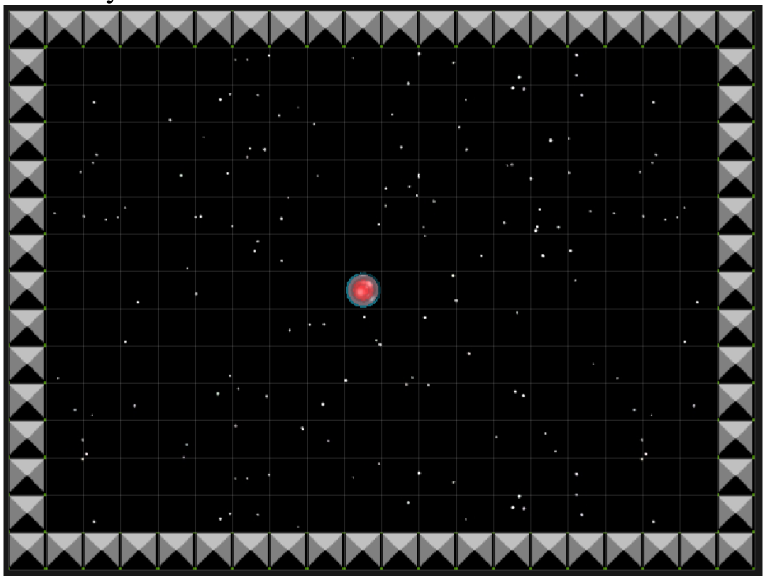
Draw 4 walls around your room, enclosing it.

Next, place one of your button objects near the middle of the room.

The image on the next page shows a completed scene

Now, if you run your game, you should get something that looks very much like your room, but nothing is happening. To be able to interact with objects, you need to program into them events and actions.

Close the room tab when you are done.



Stage 7. Events and Actions.

These define the actions and reactions of each object you create. They, in a sense, turn scenery into a game that you can play, but require a bit of thought. Careful planning (or lots of experimentation and a bit of luck) is needed to ensure that the objects perform the way that they are supposed to. So what is this game going to do?

1. The object should be able to move.
2. The object should be able to bounce off walls but is very hard to do without know much more about coding in Game Maker. So instead we will make the button move to a random position in the room and then move in a random direction.

Then, once that is done…

3. When the object is clicked on by the user, it reacts.
4. The object should move randomly after a set amount of time.
5. The game should keep track of the player's score.

So, now that there are specific ideas that need to be implemented, a look at some of the tools that are available are needed. As usual you can find more detail in the help file.

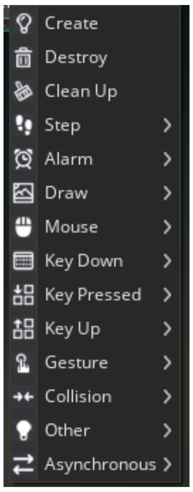
First of all, the concept of events and actions need to be explained. In each object property window, you will see two empty areas (panes). The 'Events' pane, and the 'Actions' pane. An event is when something happens. When an object is created, that's an event. When there's a collision between two objects, that's an event. When you press a mouse button, that's an event. The actions are what the game should do when an event occurs. Things like moving, changing direction, destroying the object, making it change to a different sprite or object. They're all actions.

With the list of 4 things that we want to do, we need to think about the types of event that are needed with each. Look at the first one.

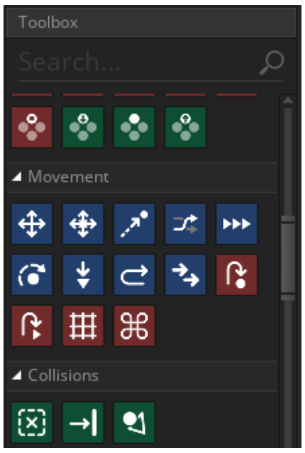
1. The object should be able to move.

But when does this need to happen? Do you want the object to start moving after a set amount of time? When the user interacts with the object? Well, for the moment, make it so it starts moving when the game starts.

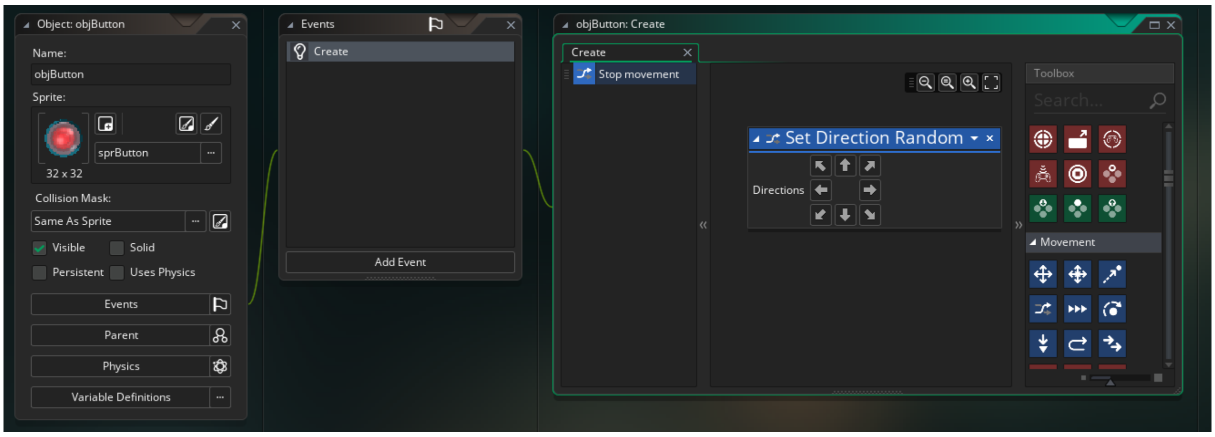
- Double click on your button object, to bring up the object properties window.
- Under the events pane, click on the button **Add Event**.
- A popup window will show up with all the defined events that you can choose from. Click on **Create**.



You will see the Actions window appear for the Create event. Movement actions can be found in the Movement group. Scroll down the toolbox until you find the Movement Group.



This is the movement button, hold your mouse pointer over the button and you will see it is called Set Direction Fixed. When placed as an action, it will make the object move in the defined direction when the event occurs. But our button should move in a random direction chosen from a set of directions. Find the Set Direction Random button, , and drag it into the actions window and you should see the following:



Name: objButton
Sprite: sprButton 32 x 32
Collision Mask: Same As Sprite
Visible ✓ Solid
Persistent Uses Physics

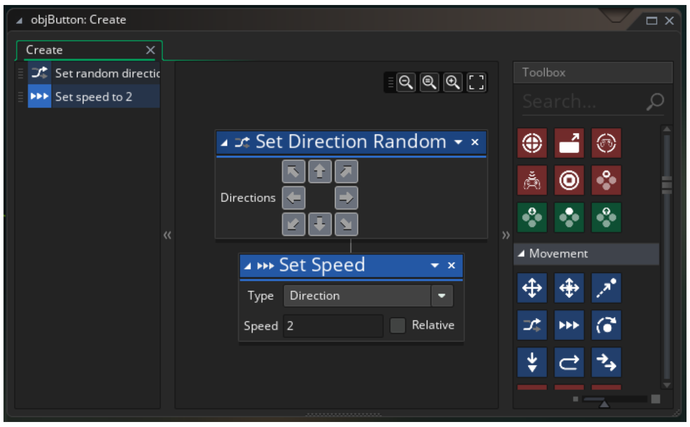
What this means is that when the object is created, some or all of the actions (of which there are not yet any) in the ‘Actions’ pane will be performed. All objects are created when the level is loaded/started, so this event will trigger when time the game starts.

What needs to be done is to choose the speed and the direction to move the object in when the game starts. You can choose any number of directions simply by clicking on any of the arrows. You can press more than one of them down, and the object will move in a direction randomly chosen from those you press down.

- Set the event to choose a random direction from the 8 directional buttons.

Then drag the Set Speed () action into the Actions window.

- Set the speed value to 2 pixels per step. Don’t tick the Relative option.



Type Direction
Speed 2 Relative

Run the game, and you should see your object move off in a random direction, and disappear off the screen. Since the collision event hasn’t been created between the wall and the object yet, nothing happens. That’s what needs to be done next!

Test the ‘game’ at this stage. Experiment with the direction settings. Save your game. Game maker saves all the information associated with a game to the folder created for you when you create a new game.

Session 1: REVIEW PAGE **Name:** MIN SOE HTUT

Questions: Complete the following questions and be prepared to demonstrate techniques to your demonstrator. (You can then have your work verified.)

1. What is a 'room' in Game Maker?

It is like a level in a game where your game take place.
Where you can place background, object and layout to create a game.

2. Everything seems indirect. You 'create' a 'background sprite' and then associate it with a room later. Surely it would be simpler just to choose a picture to use as the background from within the room editing window and not bother with 'background's as sprites at all. Why do you think the program designer set things up as they are?

By creating background sprites separately if you want to change background, you only need to modify or replace the background or sprits, object without alerting the room, it makes the layout more flexibility organize and efficient.
By breaking down a game into smaller interchangeable parts it can create a consistency across multiple room.

3. If you didn't put the Set Speed action in the Create event what happens? Why do you think this happens?

   The obj Button will not move, there will be a move event but without speed the button cannot move.

4. The background sprite is not very big. How does the whole background of a room get filled in?

   By ticking/changing Heroization title and Vertical title it will extend to the end of the grid

Verification: To assess your competency of this material your demonstrator will verify that you have completed the above questions, and can do the following:

1) Add a new room and set a background
2) Add a new Sprite
3) Add a new object using your new sprite
4) Make the object move horizontally when it is created with a speed of 10

Games – Clicky to Space Invaders – Session Two
Events: Finishing the Clicky game

In the last session you

- Created a room/level and decorated it with a background.
- Created or loaded sprites and using those, created objects.
- Added a first event to get an object moving

At the end of this session, you should be able to…

- Understand more kinds of events
- With all the above, create a complete game.

If you have shut it down, restart Game Maker and reload the game so far.

When your game opens the Game Maker screen will look as though there was no game there – nothing you created will be visible. This is just because the folders are closed. If you click on the little ▷ icons to the left of **Sprites**, **Objects** and **Rooms** the folders will open and your objects will be visible.

The first event satisfied the first objective in the following outline of the action requirements of the game.

1 The object should be able to move.
2 The object should be able to move to a random position.
3 When the object is clicked on by the user, it reacts.
4 The object should move randomly after a set amount of time.
5 The game should keep track of the player's score.

Now: The object should be able to a random position.

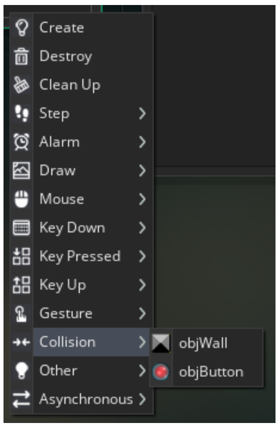
This is the second objective, and can be achieved in a similar way to the first objective in terms of the events and actions. What you do is add a new event that tells the game to do something when there's a collision between the wall and the object. The theme here is

If something happens, the game should react by doing ...

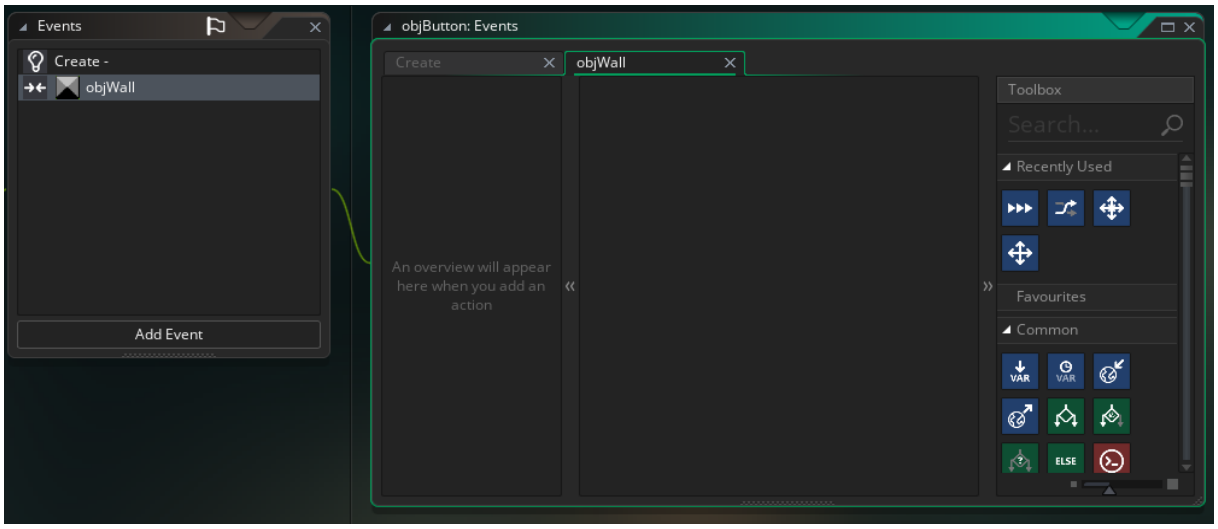
The *something* in GameMaker terms is an event. The response is one or more *actions*. To program events we need to answer another question first: *Which object is associated with the event*. We need the answer to this question because that is the way that Game Maker organises a game. We don't just program a list of events – rather we associate events with objects. In the case of a collision there are two objects involved: the moving one (the button in our game), and the wall object. We will start from the viewpoint of the button.

Double click on the button object to open the button object properties window.

Add a new event, choose **Collision** as your event type. A little pop-up will appear asking you to choose between **objWall** and **objButton** – you are being asked *What kind of object is the second object in the collision*. Click on **objWall**.

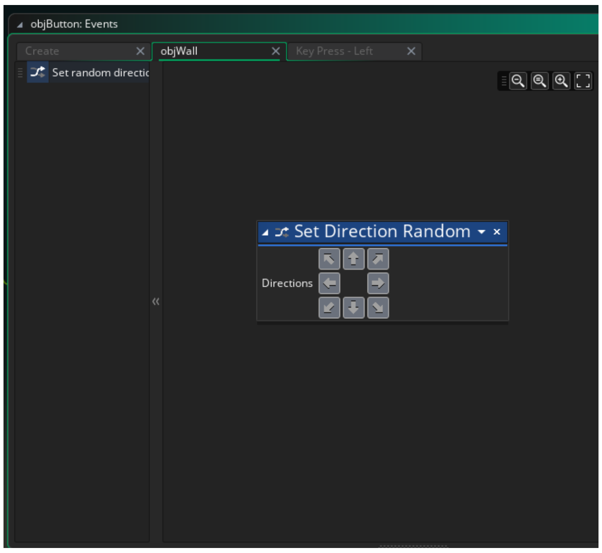


You should see something like this.



Now you have an event that happens when the button object collides with any wall piece that you have placed in the room. Now all that needs to be done is add an action to it.

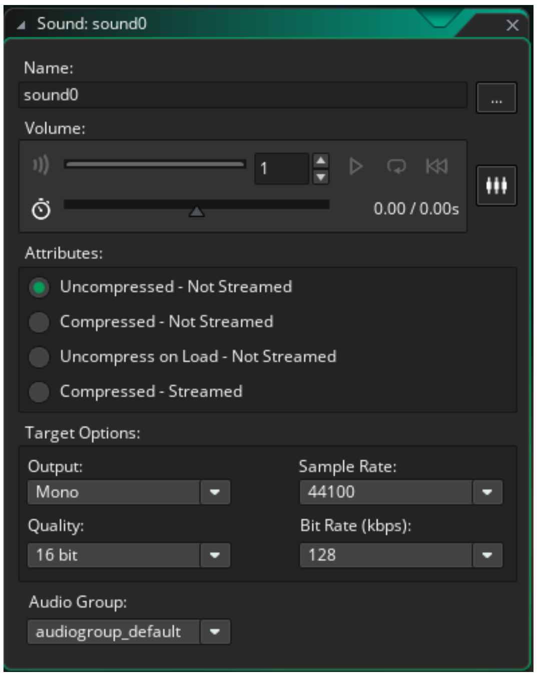
Set the direction to a random direction like you did earlier. The code should look like this:



Run the game again. This time, the object should start off and when it hits a wall it will bounce off. It isn't perfect but it will do at the moment. Now, how about some sound?

You will find in the GamesClicky folder a sound file called "bounce.wav".

First, to use a sound, you need to tell Game Maker that it exists for this game, as you did for the sprites and backgrounds. Right click on the Sounds folder in the Resources window and select Create Sound.


Sound: sound0

Name: sound0

Volume: 1

0.00 / 0.00s

Attributes:
- Uncompressed - Not Streamed [✓]
- Compressed - Not Streamed
- Uncompress on Load - Not Streamed
- Compressed - Streamed

Target Options:

Output: Mono

Sample Rate: 44100

Quality: 16 bit

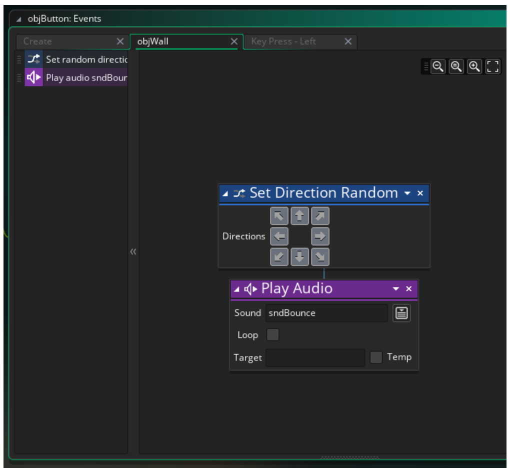
Bit Rate (kbps): 128

Audio Group: audiogroup_default

Here you can load or save a sound, and when you have a sound loaded, you can 'edit' it. Click on the '…' button next to the name textbox and load the "bounce.wav" file from the GamesClicky folder. Change the name to 'sndBounce', and to hear the sound, press the play button.

Close the sound properties window, and now you're ready to put this sound effect into play. Open the Object Properties window for your button object (if it is not still open), and select the event for the collision with the wall. Go to the Audio group in the toolbox and drag the play audio action into the actions window.

Nice and easy. The "sound" field is the sound you wish to play when this event occurs, and the "Loop" tells the game to either play it once (false) or play it continuously (true). Change the "sound" option using the button next to the "No sound" field, and choose your sound to play. Press OK, and try your game again. If all goes well, you will hear the sound play every time your object hits against a wall.



Sound sndBounce
Loop
Target
Temp

Aside: Points to ponder. You should try some experiments to find answers to these questions, or just to try some variations. It is difficult to learn by just following instructions. In the interests of continuing to make progress on your game it will be best to make a new file (you can copy and paste the game folder or use **File > Save As**), or several new files for experimenting. You will be asked questions about these in the review sheet.

Ponder 1: We installed the bounce action as an event belonging to the button. However, a collision occurs between two objects – could we have started with the wall, and added a collision event to the wall object? [Hint: The answer is yes] What would we have to do to make it work?

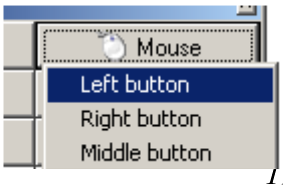
Ponder 2: The collision we defined dealt with collisions between buttons and walls. Can we define a collision between buttons? Try starting the game with one, a few, many buttons and set up collision events between buttons.

Interaction

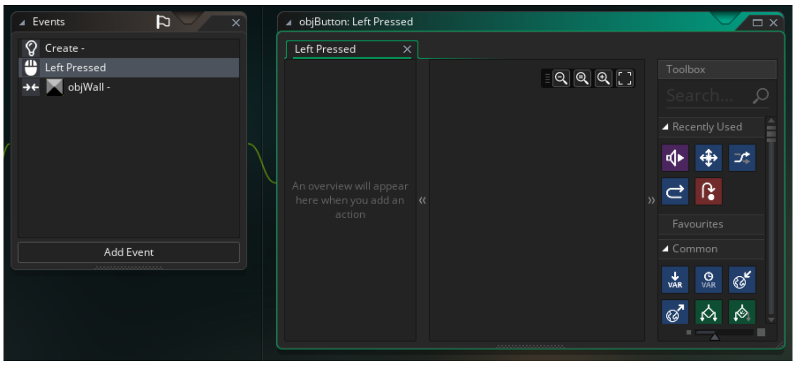
Time to put in what makes a game a game – being able to interact with it. So far, all that you have is an object that bounces against walls. To take that a step further you'll want to put this in.

3. When the object is clicked on by the user, it reacts.

For this, there's an event you can put in your object to be fired when the user clicks on it with a mouse button. Add a new event in the

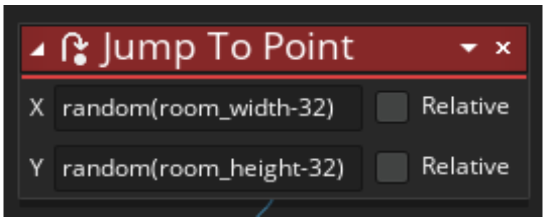


Object Properties for your ‘Button’ object. It’s located under “Mouse” in the event selector. Choose the “Left Pressed” option, and you’ll get something like this.



Now that you have an event that occurs when you click on your object with the mouse button, you get to choose what will happen. Maybe turn invisible for a short while? Go faster? Jump to a random position? Make it move in a different direction? Well, how about implementing the last three.

If you look in the Movement group in the toolbox you will see the Jump To Point action (), drag it into the Actions window. We now need to set it to jump to a random x and y position, type in the following:

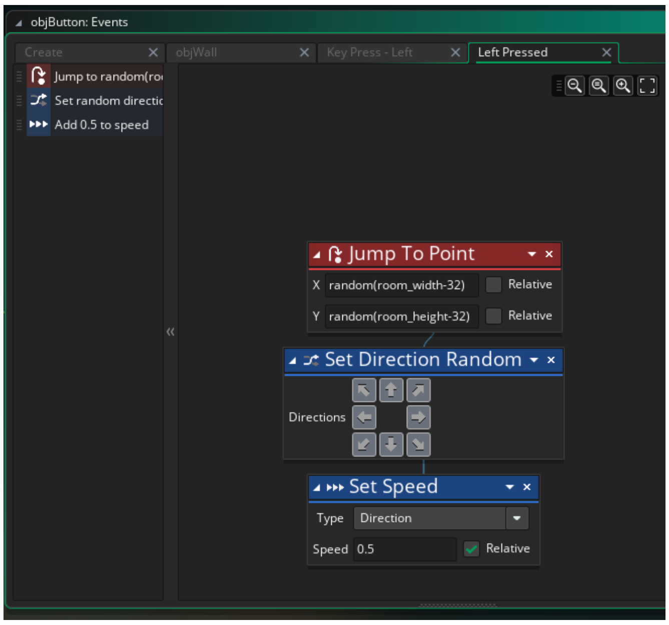

Jump To Point

X random(room_width-32) Relative

Y random(room_height-32) Relative

This means the x position will be a random value between 0 and the width of the room (subtracting the width of the sprite) and the y position will be a random value between 0 and the height of the room (subtracting the height of the sprite). Also then make the button start moving in a random direction as well.

Now, you need to make the button move slightly faster. In an earlier part of this exercise, you made it move at a set speed. In the “Left Pressed” event, add the “Set Speed” action but instead of setting the speed to 2, set it to 0.5. Also, tick the relative checkbox on. What this means is that whenever you click on the object, it will jump, and then it will choose a new random direction, and since the relative box is checked, the value in the speed box will be **added** to the current speed of the object, making it go faster. If you made it -0.5, it would go slower. You should have something looking like this.



objButton: Events

Create | objWall | Key Press - Left | Left Pressed

- Jump to random(ro
- Set random directic
- Add 0.5 to speed

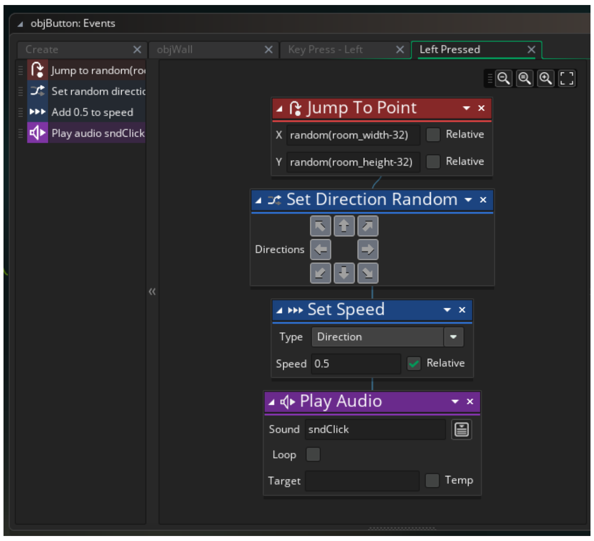
Jump To Point
X random(room_width-32) Relative
Y random(room_height-32) Relative

Set Direction Random
Directions

Set Speed
Type Direction
Speed 0.5 ✓ Relative

Test your game again. Click on the object and if you hit it, it should jump to a new position and move in a random direction, and be a bit faster. You may have to click 3 or 4 times before you are sure that the speed is changing.

One more thing you might want to include is a sound effect when you click on the object successfully. Do you remember how you added the bouncing sound when the object hit the wall? Do the same again. Add the sound file ‘click.wav’ from your resources. Then, once you’ve created the new sound, add an action to play it in the actions box for the “Mouse Button” event. Now, when you play it, every time you click successfully on the object, it will move, change speed and direction, and will also play a sound. If you’re unsure, or you are having problems, you should check to see if you have something looking like this…



objButton: Events

Create | objWall | Key Press - Left | Left Pressed

- Jump to random(ro
- Set random directic
- Add 0.5 to speed
- Play audio sndClick

Jump To Point
X random(room_width-32) Relative
Y random(room_height-32) Relative

Set Direction Random
Directions

Set Speed
Type Direction
Speed 0.5 ✓ Relative

Play Audio
Sound sndClick
Loop
Target Temp

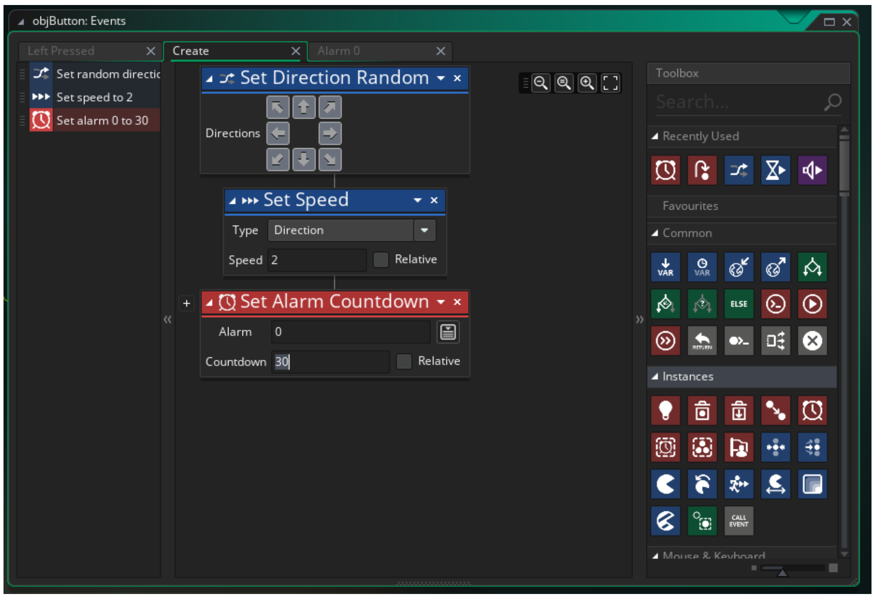
Now, for task number 4. We add this to make the game play less predictable. How did you play when testing the game? The obvious thing to do is to place the mouse ahead, in line with the object’s movement, and wait until it passes. We can reduce (but not completely destroy) the effectiveness of that strategy by making the object move randomly from time to time.

4. The object should move randomly after a set amount of time.

In this piece of the module, you will be introduced to timers and alarms. When you want something to happen after a set amount of time, an alarm needs to be used. An alarm is like a kitchen timer. You set it, and after the set time it goes off. In Game Maker terms, an alarm *going off* is, of course, an event. So, to make something happen after a time, two things need to be done: the alarm must be *set,* and an event must be arranged to do something when the alarm goes off. Actually there is a third thing to do also. If we want the event to happen a second, third … time we must set the alarm again (reset it) after it has gone off. But firstly, setting the alarm.

This is usually done in the “Create” event of the object that is interested in the alarm. Game Maker has 12 alarm clocks that you can use (numbered 0 to 11). This is a limitation of the system. If a complex game needed more timing that 12 alarms could accommodate, you would need to find some other way of solving the problem. None of the alarms will activate till they are set, by initializing them to a value greater than 0.

Open your Object Properties editor for your button object, and go to the “Create” event. Find the Instances group in the tool box and drag the ‘Set Alarm Countdown’ action () into the create event. You should get something like this show up.


objButton: Events

Create event actions:
- Set random directic
- Set speed to 2
- Set alarm 0 to 30

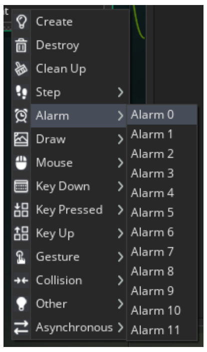
Set Direction Random
Directions

Set Speed
Type: Direction
Speed: 2
Relative

Set Alarm Countdown
Alarm: 0
Countdown: 30
Relative

If you remember back, in an earlier part of this module there was mention of “steps” in regard to how fast the game went. Here you have a field called “Countdown”. This is the number of steps the game must perform before the alarm is triggered. Game Maker games work in steps – the picture on the screen is updated at certain number of times per second. These updates are called *steps*. If the game is updating at 10 steps per second, then 5 steps represents half a second. The update rate (steps per second) can be changed for each room, but at the moment, we have left it at the default of 30 steps a second. A step is therefore about 33 milliseconds.

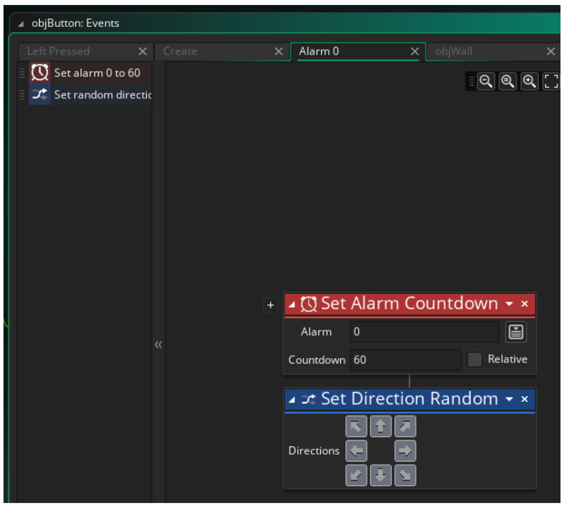
The number of seconds you want to wait before making the object change direction is up to you, but somewhere around 2 seconds or 60 steps is a good value. We can stick with alarm number 0.



Every time that alarm reaches 0 (when 60 steps have expired), it causes an event to happen. What needs to be done is for you to capture that event and use it. Add a new 'Alarm' event, and add "Alarm 0" like so…

You will get an "Alarm 0" field in the Events window. This event will happen every time the alarm reaches 0, but will not reset, so the first thing that needs to be done is to make sure that the alarm is reset to 60 steps. Drag in the 'Set Alarm Countdown' again, and set the alarm back to the value you initialized it as. This means that every time the timer expires and triggers, it will reset and begin to count down again. Now that you have your timed event going, you need to do something with it.

The objective of this event was to make the object change direction every 2 seconds or so. You have the 'every 2 seconds or so' part, so all that needs to be done now is to make the object move in a random direction. Remember right back at the start of this exercise, when you first started making events, you made the object start off moving in a random direction? That needs to be done again.



objButton: Events

Left Pressed | Create | Alarm 0 | objWall

Set alarm 0 to 60
Set random directic

Set Alarm Countdown
Alarm 0
Countdown 60 Relative

Set Direction Random
Directions

Test your game. Just watch it, and you should see the object change direction every two seconds (or however long you specified).

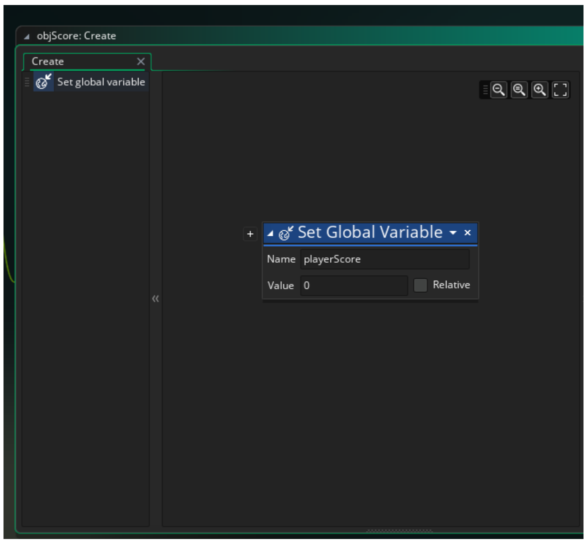
Last, but not least, the player needs to have some idea of how well they're going. Introducing the idea of a score to this game gives it a competitive aspect, and a reason to play it. For this particular game, adding a score is simple.

5. The game should keep track of the player's score.

Gamemaker has a built in scoring system but for some reason it is not working correctly with the drag and drop projects. You will learn how to create your own scoring system.

Create a new object called "objScore" and tick the persistent option but don't give it a sprite. This means when you add the score object to your first room and it will be present in every room thereafter and will keep the current score as you move from 1 room to another room.

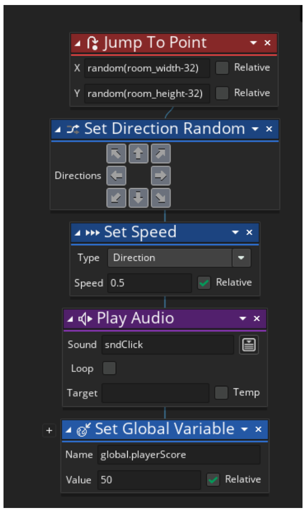
Add a "Create" event to the score object and you will create a global variable to keep track of the score. A global variable is one that can be accessed from outside the score object so that we can access the current score when required. Drag in the "Set Global Variable" action and type in "playerScore" as the name of the variable and its value should be set to 0.



Set Global Variable
Name playerScore
Value 0 Relative

Add one score object to your Level1 room. It should have a small question mark instead of a sprite.

Go into your "Left Pressed" event and drag the 'Set Global Variable' action into the actions area. For the name of the variable to add to type in "global.playerScore". Because this action is not inside the score object you must put global in front of the playerScore variable name. Put in a value that you think the player deserves for clicking on the object. You must also tick the relative button so the value is added to the score. What do you think will happen if you don't tick the relative button?



Jump To Point
X random(room_width-32) Relative
Y random(room_height-32) Relative

Set Direction Random
Directions

Set Speed
Type Direction
Speed 0.5 ✓ Relative

Play Audio
Sound sndClick
Loop
Target Temp

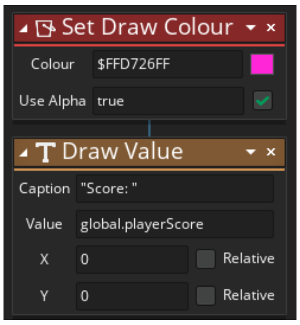
Set Global Variable
Name global.playerScore
Value 50 ✓ Relative

Test your game. What happens when you click on the button? To see the score you must draw the score in the room.

Open your button object and click on the "Add Event" button and then click on the "Draw" option, from the pop out menu select "Draw Gui". You can display the various parts of the game interface in this event, like showing the score, the number of lives, amount of ammo the player has, etc. The Draw Gui event is executed many times per second and will constantly update the state of the game.

Find the 'Drawing' group in the toolbox and drag the 'Set Draw Color' action () into the actions area. Click on the white box next to the colour field and choose a nice bright colour that will contrast with the black background. Then drag in the "Draw Value" action (). Change the caption to be "Score: ".

The x and y fields allow you to position the score in the room and the caption field is what will appear before the actual score. Leave these as the default values and your code should look like this.

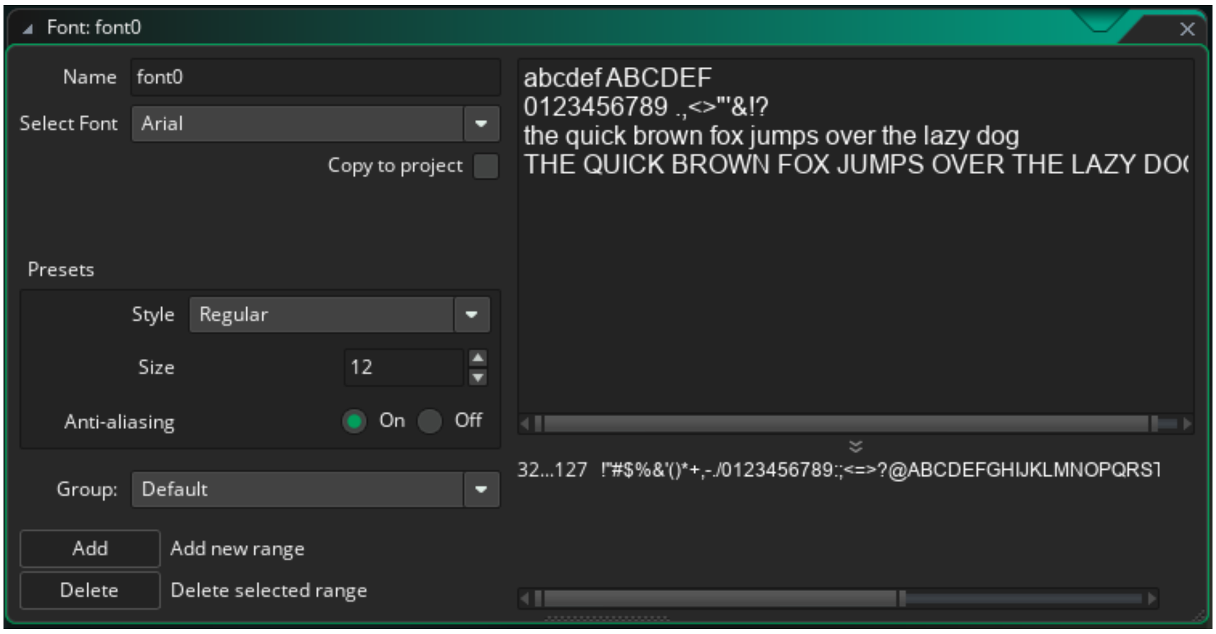


Set Draw Colour
Colour $FFD726FF
Use Alpha true ✓

Draw Value
Caption "Score: "
Value global.playerScore
X 0 Relative
Y 0 Relative

Test the game. Notice that the score appears in the wall. Go back to the Draw score action and change the x and y values to 32 x 32. Run your game to see what it looks like.

The size of the text is quite small, to be able to use larger text and different fonts you must first create a font for your score. Right click on the Fonts folder on the right hand side and select "Create" and then select "Font", and the following window will appear:


Font: font0

Name: font0

Select Font: Arial

Copy to project

Presets

Style: Regular

Size: 12

Anti-aliasing: [✓ On] Off

Group: Default

Add — Add new range

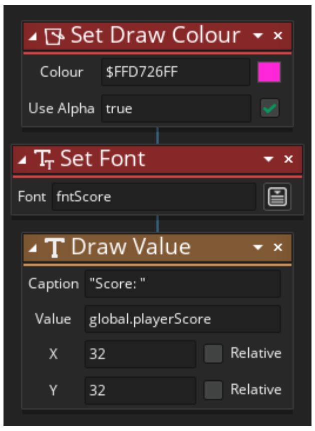
Delete — Delete selected range

abcdef ABCDEF
0123456789 .,<>"'&!?
the quick brown fox jumps over the lazy dog
THE QUICK BROWN FOX JUMPS OVER THE LAZY DO

32...127 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRST

Rename the font to “fntScore” and then choose any font you want from the drop down button. Set the size field to around 16 or 18 and feel free to try different fonts if you wish and then close the Font properties window.

Go back to the Draw Gui event in your button object and from the “draw” tab drag the Set font action () before the Draw Value action, then in the window set it to the font you created and click on “OK”. Your Draw Gui event should look like this:


Set Draw Colour

Colour: $FFD726FF

Use Alpha: true

Set Font

Font: fntScore

Draw Value

Caption: "Score: "

Value: global.playerScore

X: 32 Relative

Y: 32 Relative

Run your game and you should see the score displayed in your chosen font.

Extras

To have your session verified you should complete at least one “extra”.

Now that you have your game up and going, you can think about what you might want to add to it. You can experiment by putting in more walls in the actual playing field for more interesting movements, changing the speed that your object moves, or even add in another of your objects and have the two of them bouncing around at the same time.

One other thing you might want to do is add background music. You can add a wav or mp3 file to your game like you did earlier. To play this music, you need to set it playing in an event like any other action, but this is a little tricky. Normally, in a game, there is a controller object that you cannot see when you play the game. This object takes care of all the events and actions like playing the right music at the start of a level, keeping track of areas of your level you can't see and so forth. Since we haven't created one for a game this simple, the start music event needs to be placed on something that is always going to exist, but of which there is only one. If your game has only one 'button' object, you can put it on that; or you can create a new object with no sprite and put that in your room. For now we will assume that there is only one 'button' and put it on that.

Open the object properties for your button object, and go into the "Create" event. Drop in the 'Play Audio' action, and choose your new music file that you've just created. Now, before you press OK, you'll want the music to loop, so turn tick the 'Loop' option.

Play your game! You should have music in the background, sound effects when your object bounces against the wall and gets clicked on. It's not the next World of Warcraft, but it demonstrates many of the basic principles of game construction.

NOTES

Mouse events. You may have wondered why the mouse event you added fired only when the mouse was over the button. There are more general mouse events – the *global* ones — that fire regardless of the position of the mouse.

Sound editor. At some point we noted that there was an option to 'edit' a sound. Game Maker does not come with an integrated sound editor. It simply loads the system recorder accessory, which offers little in the way of sound editing. If you need something more sophisticated look at the COMP123 Sound Editing module.

Session 2: REVIEW PAGE **Name:** MIN SOE HTUT

Questions: Complete the following questions and be prepared to demonstrate techniques to your demonstrator. (You can then have your work verified.)

1. If the collision between button and wall is defined from the point of view of the wall, how do you make the collision work correctly? (Ponder 1)

   If the button hit the wall the button will bounce back the ball (it is not really bouncing it just stop and move to different direction), the ball is not bouncing, the wall is making the ball the bounce back.

2. Why is it better to do it from the button's perspective?

   Because it can be used to control the collection between two balls. the collection animation is more smooth.

3. Can you think of any reason to do collisions from the wall's view point?

   No
   It may be easier to manage if you have different kind of balls but I think the result will not be that different.

4. (Ponder 2) Did you discover anything about collisions here?

   Yes, when the three or more balls are in the same space the collection become chaotic.

5. What happens when you don't tick relative when adding score if the player clicks on the button? Explain why this happens?

The Count will not go up because the counting is not editing the new count into previous one it will just be displaying the current count withing editing.(i += 1)

6. Which extra(s) did you add to the game?

   More Walls and Buttons
   And Background Music

Verification: To assess your competency of this material your demonstrator will verify that you have completed the above questions, and can do the following:

1) Add a right click even to the button object that will make it jump to a random position, change direction, reduce speed 1, play click sound and add 100 to the score.

Games – Clicky to Space Invaders – Session Three
Controlling a player: Starting Space Invaders

This session starts the build of a Space Invaders game. The major new features are controlling a play piece and firing weapons. Lots of detail arises on the way. A rough table of contents for this session follows.

- Name the game, and explore the parameters.
- Create sprites and backgrounds.
- Scrolling background.
- Put the character on screen.
- Illustrate the different ways of generating movement, and then use the mouse movement.

In this second half of the module, it is intended that you will be able to create some more advanced event handling, understand in greater depth the way a game can work and some of the underlying mechanics that need to be taken care of to make sure your game runs smoothly.

The first thing to do though, is understand a bit more about how the games work in Game Maker. Open Game Maker, and create a new Drag and Drop project. You will only have one room for this example. The first thing to learn is how to name your room, and make the name appear in the title bar when your game is running.

Name the game, and explore the parameters

Change to the 'Settings' tab in the room properties window. Set the name of the room to "Space_Invaders". It is a good idea to give your rooms proper names which makes it easier when making changes to games with lots of rooms.

Let's have a look at some of the other settings here. Width and Height are the dimensions in pixels of the play area. Generally, the larger those numbers, the slower the game will go on older machines, so change the width to 640 and the height to 480. Speed is the number of steps per second at which your game is updated. Here you can set your game to update very fast or very slowly. For some games (like Boulderdash for instance, where you can only move about 4 or 5 steps per second) it is advantageous to have a slow speed setting for the rooms. In this game, with continuous action, a faster speed gives smooth game flow.
30 steps per second is a good value.

The tiles and views tab don't need to be looked at yet, but if you're interested, the views tab allows you to set a portion of the room to be visible at one time, and tiles allow you to paste graphics on static, non-moving objects as a way of increasing performance. Both are used quite frequently in creating platform (side scrolling) games, but neither is going to be used in this tutorial.

Create sprites and backgrounds

For each sprite you create, leave the default values alone. Just load them up and ok them.

In the resources folder on L: (COMPX251\GamesInvaders), you'll see a file called "Player Ship.gif". Load this file as a sprite and give it a suitable name (remember to put 'spr' at the beginning of the name), then edit the sprite and remove the background colour. Then click on the arrow next to Collision Mask to drop down it's options. Change the 'Type' setting to 'Precise'. Precise collision checking means that collision will occur precisely with the outline of the sprite, without it collisions will occur with the sprite rectangle of 32 x 32 pixels even if there is empty space between the bounds of the rectangle and the actual sprite image.

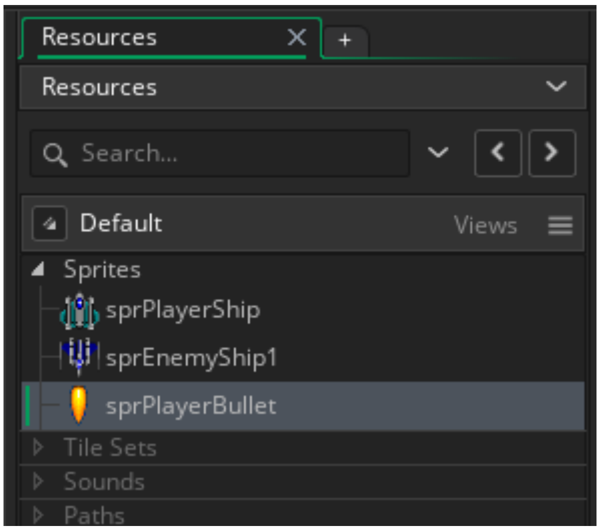
To see the difference tick the Preview Mask button and swap between Rectangle and Precise to see the differences. Make sure you leave the setting as Precise.

With the Precise setting, in the preview you can see that only the sprite image is darkened. With the Rectangle setting you will see the whole rectangle darkened which means you can have a collision with empty space. For our game it is better to set it to Precise.

This tutorial will be calling it the "Player's Ship", so when you see those words, you know what we're referring to.

Create and load another sprite using the file "Enemy Ship 1.gif" and remove it's background colour and set the collision type to Precise.

Lastly, create a third sprite, load "Player Shoot.gif" and give it a name (sprPlayerBullet) and remove it's background colour and set the collision type to Precise. You should end up with something looking like this.



Sprites
- sprPlayerShip
- sprEnemyShip1
- sprPlayerBullet

Tile Sets
Sounds
Paths

Scrolling background

Create and load a background sprite. Load the file called "Stary Background.bmp".

Now go back to your room and place the background in the room and tile the background to fill the entire room.

Next, to do something a little more fancy. Here, you can move the position of the background by changing the X and Y values, but the more interesting option is being able to make the background scroll!

Scroll down the Background Layer Properties section and find the 'Vertical Speed' option. Set the Vertical Speed to 1 in the background tab for your room.

Run your game again and notice what's happening. Your background is now scrolling down the screen. By changing a single number, you now have a moving background. Experiment with different values for the vertical and horizontal speeds of the background, and when you're finished, set the vertical speed to 1, and the horizontal speed to 0.

Put the character on screen

Now that you have the scrolling background, you need to put your ship on screen. Create a new object, and give it the Player's Ship sprite. You don't need to make it solid, as it's not going to be used to restrict anything. In your room, place an instance of your ship (place a single ship object in the room). At the moment it does nothing – it's time to look at the different ways to make the ship move.

Illustrate the different ways of generating movement

First, you have to think of the ways that you can interact with the game. You have a keyboard, and you have a mouse. Either could be used to control an object.

For the keyboard there are several ways of translating key presses into movement. You can press and hold a key to move left, and press another to move right. You could press a key once, and start moving left, and then press it again to stop. You could hold down that key and the longer you held it down, the faster the object would move. You could move in discrete jumps – pressing a key moving you one unit of distance.

For the mouse, there's one of two ways. Either you can move when you move the mouse, but only at a set speed; or you can make the ship move immediately to the mouse location: i.e.: follow just like the mouse cursor.

In this section you will experiment with different movement styles.

Note: You might like to save your game at this stage and make different copies for the different movement methods.

Keyboard movement version one

The style of interaction is: press and hold the left or right arrow key to move (at a fixed speed) left or right. On releasing the key movement stops.

In the player's ship object properties window you will initially need two Events. One for when you press the left arrow key on the keyboard, and one for when you press the right arrow key.

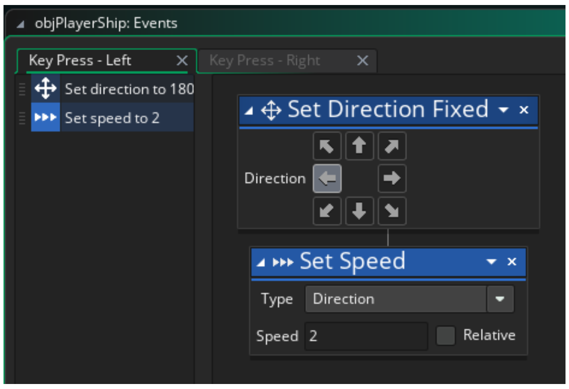
For keyboard actions, there are two kinds of events that can be generated. There are the 'Key Pressed' and 'Key Up' events, and then there's the 'Key Down' event. The first pair only trigger single events – when you press the key down, and when you release it. That is the kind we will use for this experiment. (The second type of event works as long as the key is down – a new event will be generated on each program step, while the key remains down – you can try that later.)

In the player's ship object properties window: create two 'Key Pressed' events. When you select the Key Pressed event you will see a pop-up menu, allowing you to select the key you are interested in – select <left> for one and <right> for the other to use the left and right arrow keys respectively.

Aside: Note that you could select any key on the keyboard at this point. Some games use letters (often J for left K or L for right), so that they can be used on computer with small keyboards where the arrow keys are not convenient (e.g.: some older laptops). More sophisticated games go a step further and provide a way of allowing the player to set the key assignments. We cannot do that easily in Game Maker.

As the action in each of the two events, you need to set the direction and speed of the ship.

Use the "Set Direction Fixed" action ( ) for each event, and tell it to only move in one direction (left or right as appropriate). Also add a "Set Speed" speed action and set the speed to around 2 or 3. Remember, that's 2 or 3 pixels per step, and there are 30 steps a second.


objPlayerShip: Events

Key Press - Left | Key Press - Right

Set direction to 180
Set speed to 2

Set Direction Fixed
Direction: ←

Set Speed
Type: Direction
Speed: 2
Relative: ☐

That's it for the 'on key pressed' events. Try running your program. You should be able to start your ship moving left and right, but not stop it moving. What happens if the ship goes outside the room? Can you get it back?

To stop movement, you need a 'Key Up' event. Add "Key Up" events for the left and right arrows. For each event set the speed to zero.

Test your program. You should now have control of your ship with left and right arrow keys.

Keyboard movement version two

Nice solid movements, that are easy to program, but have one drawback. What if you needed to move quickly? There's no way of making your player move quicker without mapping more keys, or changing the speed in your action. But then how would you move more slowly? You could try acceleration. Try changing the speeds in the ‘key pressed’ events to 1, set the relative flag on and see what happens.

Try it now. What happened? Why? (This is question 2 on the review page)

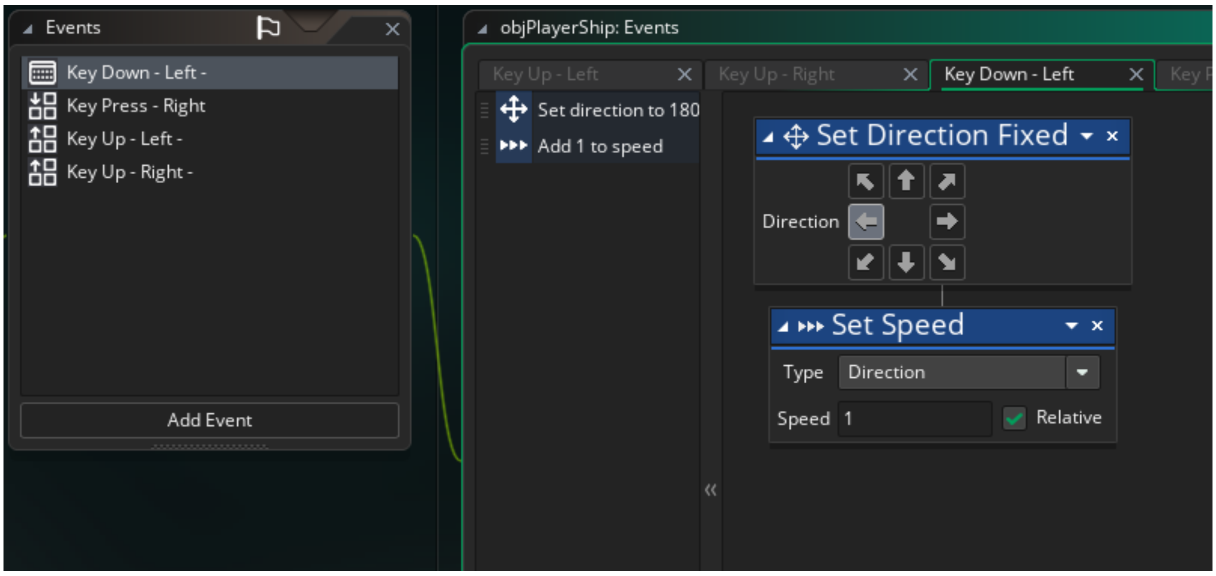
Keyboard movement version three

In this version you will try the second form of key event. To make the program change quicker you can use the following shortcut.

Right click the ‘key pressed <Left>’ event, and choose ‘Change Event’. Here, you can change the event you use to trigger your actions without losing those actions you already have there. Very handy when you have a long list of actions that you want to move from one event to another and don’t want to reprogram all those actions.

Change the event to a ‘Key Down’ event (the one without the green or red arrows on the keyboard), and choose the same keystroke.

You should now have something looking like this.



Events
- Key Down - Left -
- Key Press - Right
- Key Up - Left -
- Key Up - Right -

objPlayerShip: Events — Key Up - Left, Key Up - Right, Key Down - Left

- Set direction to 180
- Add 1 to speed

Set Direction Fixed — Direction: ←

Set Speed — Type: Direction; Speed: 1; Relative ✓

Change the ‘press <Right>’ event in the same way, and try your game out again.

What happened? Why? (This is question 3 on the review page)

You now have acceleration, and you can move slowly by tapping the movement keys, but you still don’t have the ability to flick across the screen if you really need to. Unfortunately, this is the drawback of digital input (a case where you can either have an on state, or an off state). There’s no (easy) way of letting you choose at which speed you want to go, therefore keyboard controls might not be the best for a game application such as this. You might like to investigate the original Space Invaders game and find out what movement controls it had.

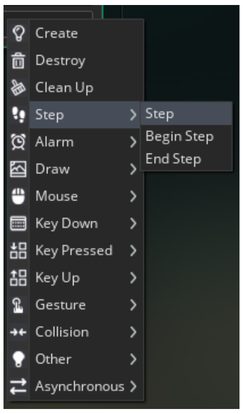
Aside: Games provide different challenges. You could argue that fighting the space monsters under the constraint of digital movement is the challenge of the game.

Mouse Controlled Movement

The mouse however, as you might have guessed by now, is the perfect candidate to be a controller for your ship.

Save the current state of your game and then “Select Project Save As…” to create a new version of the project.

Delete all the events you have for your ship at the moment (select the first event and then Shift-Click on the last event to select them all, then right click and select Delete), and then add a new event Step event. Choose the ‘Step’ option from the menu that pops up (not begin or end step).

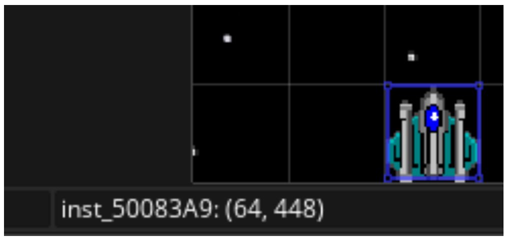


This event is one of the most powerful events that the Game Maker has to offer. This event triggers every time your game updates itself. So if your game is running at 30 steps per second, the actions placed in ‘Step’ events happen 30 times a second as well. What this means is that you can have a look at where the mouse is every step of the game. But, there’s a small problem. If you set the speed and direction of the object every step, you’ll have to figure out an appropriate mathematical expression to calculate in which direction to travel, and at what speed to move the mouse. This is rather inconvenient, but there’s another, much easier way. Why not simply jump to the current mouse cursor position?

Add the “Jump To Point” action to your Step event (from the ‘Movement’ group in the toolbox).

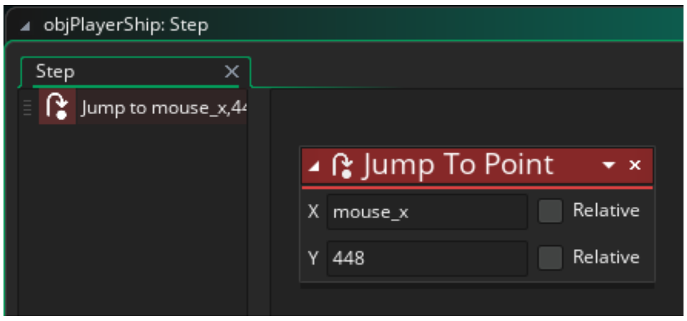
So, every step, your ship will jump to a set of co-ordinates, but you haven’t supplied any yet. Just press ok, as you don’t have all the information to set the x and y co-ordinates. Open up the room properties editor and resize it so you can see the ship that you’ve placed in there.

If you hold your mouse over the ship, you will see down at the bottom of the room properties window some information about your ship.


inst_50083A9: (64, 448)

Here, you can see it's x and y co-ordinates (64 and 448 respectively). What you are after is the y co-ordinate (448 for me). The ship in the original Space Invaders game could only move left and right; so you need to fix its y position so you can't move it up or down. So, armed with this information go back into the object editor for your ship.

Now that you have the y co-ordinate, enter that in the y field. For the x, co-ordinate, what you will want is the x position of the mouse. Thankfully, instead of having you dig around in the documentation to find out how to get this value, I can give it to you. Type in the x box 'mouse_x' without the ''. (Note the underscore character between 'mouse' and 'x'. It is needed.) You should have something like this.


objPlayerShip: Step

Step

Jump to mouse_x,44

Jump To Point

X mouse_x Relative

Y 448 Relative

mouse_x is an internal variable that the game uses to get the current position of the mouse cursor. It's always available and cannot be changed by the game, only the mouse. There's also a mouse_y, but we don't need that. Now, if you play your game, you'll be able to move your mouse over the game, and see your ship move, what happens if you move your mouse pointer outside of the room?

Session 3: REVIEW PAGE **Name:** MIN SOE HTUT

Questions: Complete the following questions and be prepared to demonstrate techniques to your demonstrator. (You can then have your work verified.)

1. If your ship moves at 2 pixels/step and you are running your game at 30 steps/second. How long does the ship take to cross the screen from left edge to right edge?

   Speed=2 pixels per step×30 steps per second= 60 pixels per second
   Time= 640 pixels / 60 pixels per second
   = 10.67 second

2. Keyboard movement version two: What happened? Why?

   The player ship is moving a faster and faster if i switch between left and right but can't stop the player ship.
   because while i was holding down the button the speed is continually increasing += 1

3. Keyboard movement version three: What happened? Why? Do you like this version?

   The player ship is moving a faster and faster if i hold down the move button and can be stop my stop pressing the movement button.

   Because while i was holding down the button the speed is continually increasing += 1 but after i stop pressing it change back the speed to 3.

   No, the movement speed can be controlled to have an efficient game play.

Verification: To assess your competency of this material your demonstrator will verify that you have completed the above questions, and can do the following:

1) Add a KeyPressed <Up> event that will move the ship up with a speed of 5.
2) Add a KeyPressed <Down> event that will move the ship down with a speed of 5.

Games – Clicky to Space Invaders – Session Four
Bullets: Firing and destroying opponents

The rough table of contents for this session is as follows

- The sprite origin, illustrated by firing weapons.
- Cleanup: destroying objects outside the play area.
- Put in an enemy entity and set up the collisions.
- Illustrate instantiation by creating multiple bad guys to shoot.

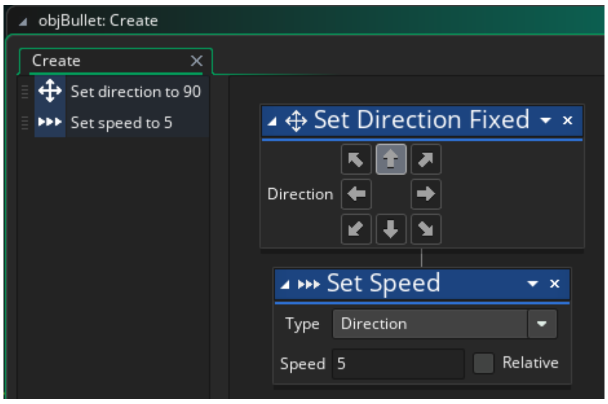
And now you move onto the exciting part. Firing a weapon! Well, currently you don't actually have any ammunition as objects, only as sprites. You will need to make a bullet object with the sprite you have already loaded. Once you have made your object, you need to make your ship fire your bullet up the screen at a certain speed.

The sprite origin, illustrated by firing weapons

Make a bullet object for your ship using the player bullet sprite you created earlier, and add a 'Create' event.

Remember, the create event lets you say what will happen at the moment of an object's creation. What you want the bullet object to do is fly up the screen, and then collisions can be considered (but those will be added when you add enemy ships). The most basic way to make the bullet fly up the screen is simply give it a direction and speed like you've done before with other objects, and just leave it.

Give your bullet actions when it's created to move up the screen at speed 5.


objBullet: Create

Create

Set direction to 90

Set speed to 5

Set Direction Fixed

Direction

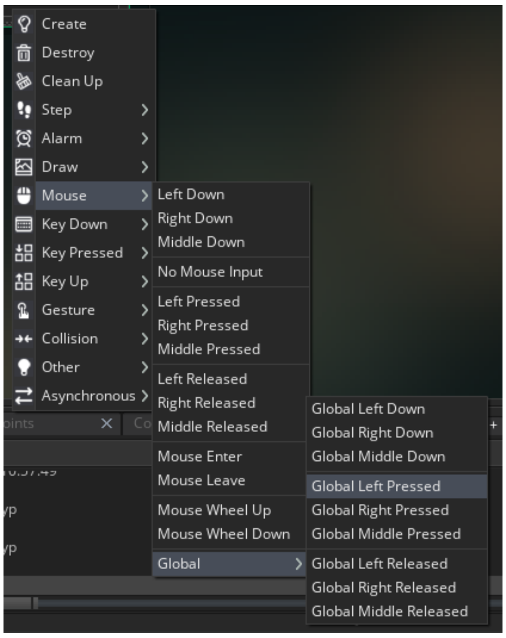
Set Speed

Type Direction

Speed 5

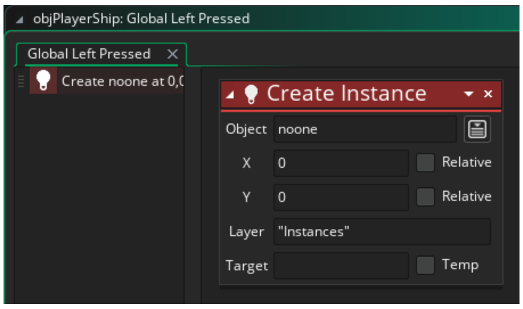
Relative

Now that you have your bullet object, you need to be able to fire it. We will have the bullet fired when the left mouse button is clicked. Open your ship object properties window, and add a "Mouse" event. The mouse event you need is called "Global left pressed" under the 'Global mouse' heading. Remember, the non global events trigger only when the mouse is over the object concerned. The *Global* events fire regardless of where the mouse cursor is, which is what we want in this case.



Now that you have the event, we simply create a new bullet near the ship. Its create event will immediately occur, so it will begin to move up the screen by itself. We use another powerful feature of Game Maker. The 'Create instance' action ( ) from the 'Instances' group in the toolbox. This creates an instance of an object at a specified position.

Add the Create Instance action to the actions window for the 'Global left pressed' event. This is what you'll see.


objPlayerShip: Global Left Pressed

Global Left Pressed

Create noone at 0,0

Create Instance

Object noone

X 0 Relative

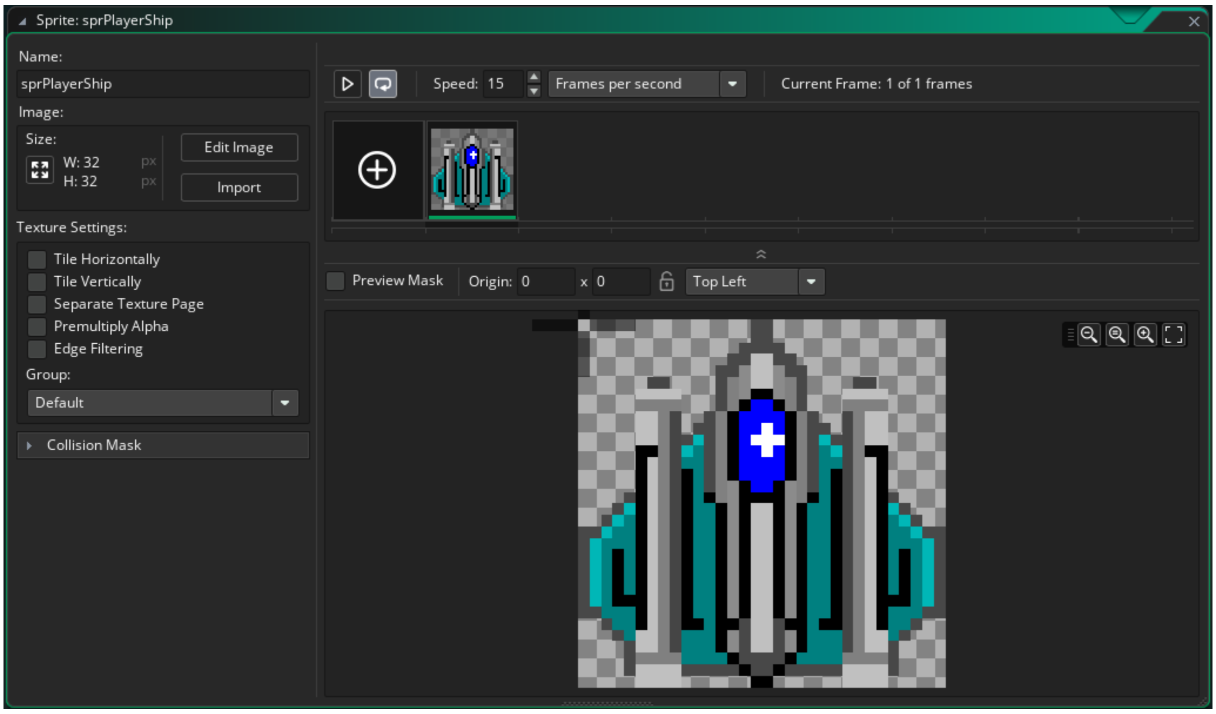
Y 0 Relative

Layer "Instances"

Target Temp

The kind of the object you wish to create is the bullet, so that's the object you want to create. From the dropdown menu to the right of the 'object' box, choose the bullet for your ship. Next, you need to choose where on the screen the bullet should appear. You'll find the 'Relative' checkbox handy here. Remember, this event belongs to the ship object, so the bullet is positioned relative to the ship. Leave the rest of the values alone for the moment, and press ok. Try out your game! If all went well, every time you press the mouse button, you should get a bullet created that shoots up the screen.

Now, no doubt, you're wondering why on earth the bullets that you're creating are coming out of the side of your ship! To fix that we will need to look at sprite properties more closely. Open the 'Player's Ship' sprite (not object) property window. It should look like this.

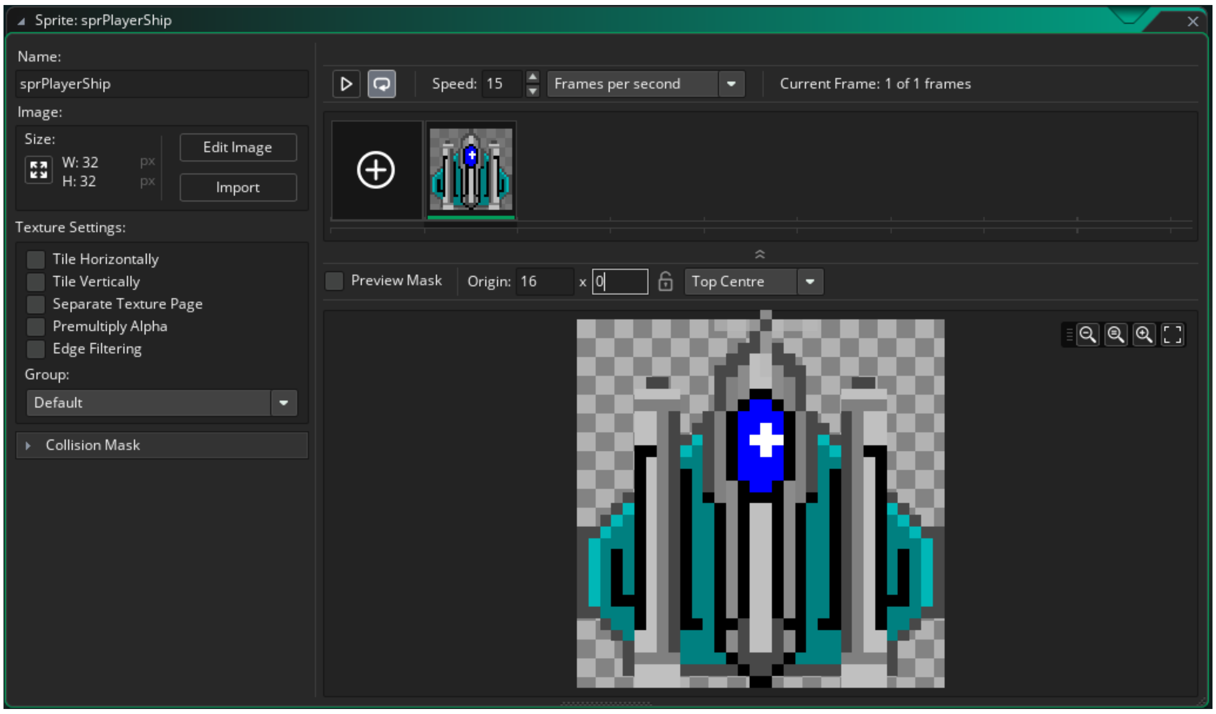


Origin. **This is the important part!** The origin x and y settings are at the bottom left of the sprite properties window. The origin settings control the placement of an object in a room. When you "Create object x at position (24, 24), what you are doing is creating an object, and putting the origin of that object at position (24, 24). By default, the origin for all objects is the top left hand corner. This applies to both the bullet and the ship. Try setting the movement speed of the bullet to zero. You should see that the bullet gets placed with its top left corner matching the top left corner of the ship. **This is why your bullets are coming out of the left hand side of your ship.**

What needs to be done to make the bullets come out of the middle of your ship is to either offset where the bullets shoot out of, relative to the ship, or change the ship's origin. We will work by changing the origin of the ship.

Change the x origin point to be half the size of the sprite's x dimension, remember the width and height of the sprite is 32 x 32. Notice the dropdown button changes to Top Centre. You can use this dropdown button to set the origin to specific locations in the sprite.

You can be out 1 pixel or so, but you'll notice in the picture that there is a line down the middle of your ship now. It's actually a crosshair, but you can't quite see it yet. You should have something looking like this.



Name: sprPlayerShip
Image:
Size: W: 32 px H: 32 px
Texture Settings:
Tile Horizontally
Tile Vertically
Separate Texture Page
Premultiply Alpha
Edge Filtering
Group: Default
Speed: 15 Frames per second
Current Frame: 1 of 1 frames
Preview Mask
Origin: 16 x 0 Top Centre

Now, if you OK that, and go into your room editor, you'll notice (or you might not, it's hard to notice really) that your ship has moved a little to the left. That's because you've changed the origin of the ship graphic, therefore, the game now shows the graphic with (16,0) where (0,0) was previously.

Change the origin for the player's bullet, move the origin into the top centre of the bullet.

Try your game out again. It should now fire out from the top middle of your ship.

You might have noticed that the bullet looks as though it is upside down. You might like to see if you can use 'Edit Sprite' to flip it up the right way.

Cleanup: destroying objects outside the play area

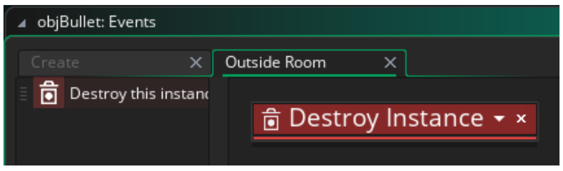
OK! Now that you have your ship, and it's able to fire, there's a small area that needs to be looked at. It's called resource management. It's something you need to keep in mind when you have things that go out of your room, like all your shots from your ship do. Ever wonder where they go? Well, they happen to keep going on and on for a very long time. Now, for most computers these days, it's next to nothing to keep control of all those objects, but it's far cleaner to get rid of them so that they don't all need to be moved each step. For this, there's another event that you'll want to use.

Open the object properties window for your ship's bullet. Add the "Outside Room" event from the 'Other' group.

This event will trigger any time any of the bullets go outside the room, and will trigger for each bullet separately (so when one goes outside, they won't all disappear.)

Now that you have an easy way to find out which bullets are going outside the room, all you need to do is destroy them.

Add the action "Destroy Instance" action ( ) from the Instances group from the toolbox.



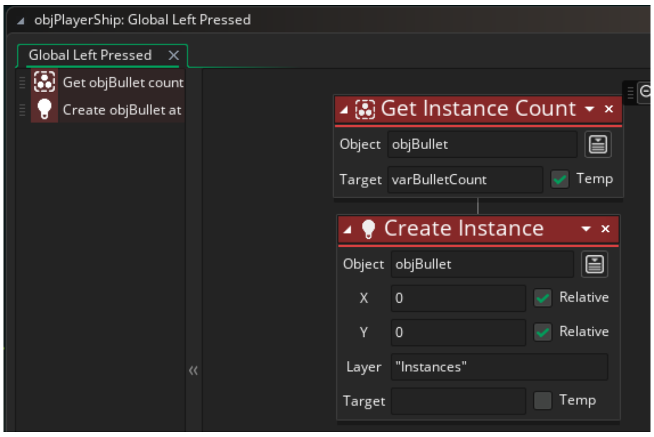
Now, you won't be able to see those bullets being destroyed in your game, but if you ran the debugger, and watched an individual bullet go out of the screen with the debugger's special tools, you'd see that it was destroyed after it left the room.

Ok, now that the resource management has been done, there's another thing that needs to be add to our space invaders clone. If you can remember the original space invaders, you could have one bullet travelling at a time. Attempts to fire a second bullet were ignored until the first bullet left the screen, or hit its target. This is the next thing you're going to implement.

Open the object properties for your ship, and go into the "Global Left Pressed" event.

Here is where you are making the ship fire, and what needs to be added is a conditional statement. A conditional statement is one that asks a question, and then performs an action if the answer to the question is 'yes'. The kinds of questions we can ask are like "If this value is x", or "When this value is y", or "While object exists". Those are conditionals. In Game Maker, all conditional actions are denoted by having a greenish background.

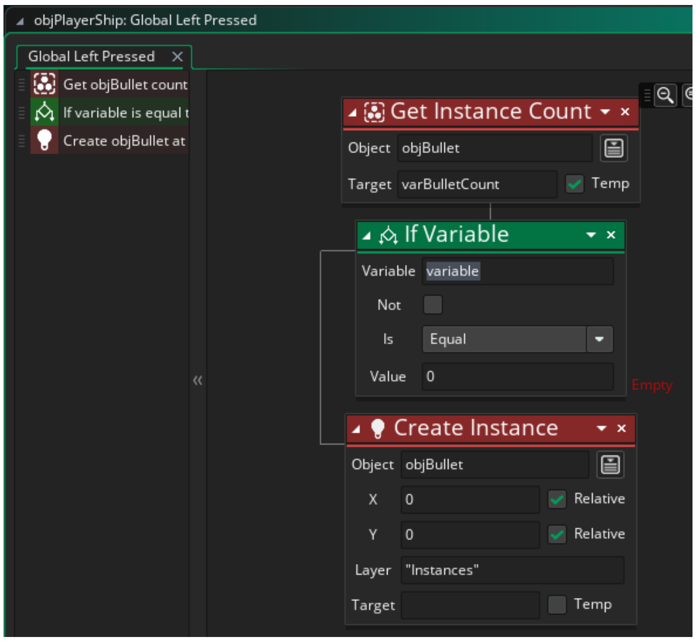
First you will need to count the number of bullet objects in the room. From the Instances group in the toolbox, drag the 'Get Instance Count' action ( ) to the top of your Global Left Pressed event. Then set the object to count to be your player bullet object by clicking the button to the right of the object textbox. You must then set a target variable which will store the number of bullets in the room. Type in 'varBulletCount' as the name of the variable and tick the 'Temp' option. This will create a temporary variable to store the count just in this event.



Object objBullet
Target varBulletCount ✓ Temp

Object objBullet
X 0 ✓ Relative
Y 0 ✓ Relative
Layer "Instances"
Target Temp

Now before creating an instance of the player bullet you need to check if the count is equal to 0. If it is then create a new bullet object, if it isn't equal to 0 then don't do anything. From the Common group in the toolbox, drag the 'If Variable' action () in between the Get Instance count and Create Instance actions.



objPlayerShip: Global Left Pressed

Global Left Pressed

- Get objBullet count
- If variable is equal t
- Create objBullet at

Get Instance Count

Object	objBullet	
Target	varBulletCount	✓ Temp

If Variable

Variable	variable
Not	☐
Is	Equal
Value	0

Empty

Create Instance

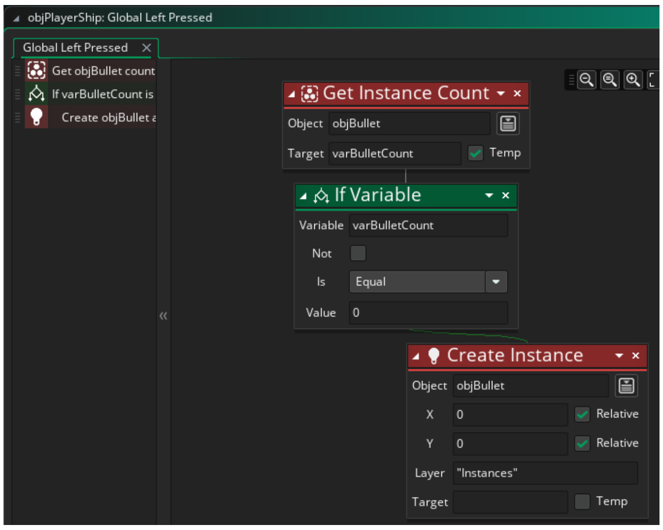
Object	objBullet	
X	0	✓ Relative
Y	0	✓ Relative
Layer	"Instances"	
Target		☐ Temp

The action following the conditional will only be performed if the conditional gives the answer 'true' – in this case that the number of instances is equal to 0. Using this action, you can also verify that your bullets are being destroyed when they leave the room. If they are not, you won't be able to fire any more shots when your first bullet leaves the screen. We can use pseudo-code (steps written in English) to describe what is happening:

```
IF the number of player bullet objects is equal to 0
    Create a player bullet object relative to the player ship
```

Can you think what you need to type into the Variable textbox? Ask a demonstrator if you are not sure. The rest of the options are exactly correct, test if the variable name that you type in is equal to the value 0. Notice the word Empty in red next to the If Variable action? It is warning you that there is no actions attached to the If Variable action if the expression is true. The Create Instance actions is just the next action to carry on with after the If Variable action. You need to attach the Create Instance action to the If Variable action.

Drag and hold the Create Instance action over the If Variable action, you will see a blue bar appear to the right of the If Variable action, that means the Create Instance action will be attached to the If Variable action. Let go of the action and you will see this:


objPlayerShip: Global Left Pressed

Global Left Pressed

- Get objBullet count
- If varBulletCount is
- Create objBullet

Get Instance Count
Object: objBullet
Target: varBulletCount — Temp ✓

If Variable
Variable: varBulletCount
Not: ☐
Is: Equal
Value: 0

Create Instance
Object: objBullet
X: 0 — Relative ✓
Y: 0 — Relative ✓
Layer: "Instances"
Target: — Temp ☐

Now it will only create a new bullet object if the bullet count variable is equal to 0.

Try your game out again, and see if your shot is being destroyed when it leaves the room, and if you're only firing one bullet.

Put in an enemy entity and set up the collisions

Now that I think I've made you wait long enough, time to put in some stuff to blow up!

Tasks to do…

• Create an enemy object.
• Set what happens to it when it's hit by the player's bullet.
• Give it a random chance to fire.

If you think you can do these yourself, give it a go, otherwise keep reading.

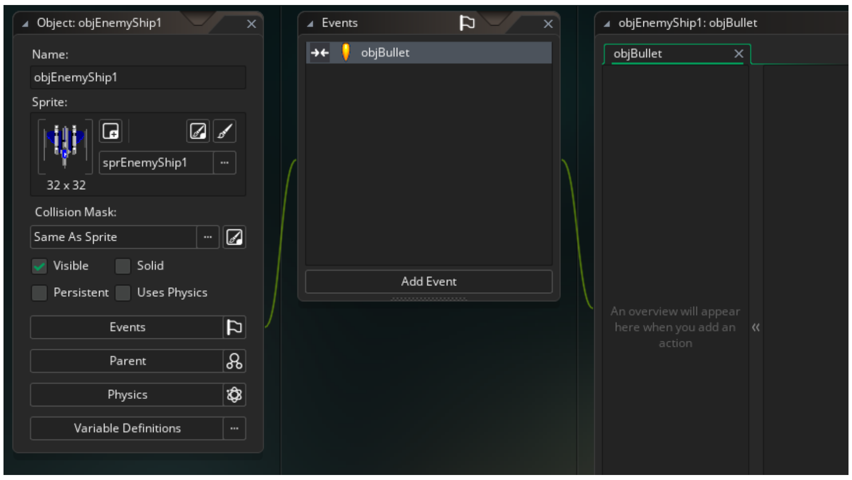
You have already loaded the enemy ship sprite, but since you know that it's eventually going to be firing bullets, you will want to change its point of origin as well.

Change the point of origin for the enemy ship to be the bottom centre of the sprite.

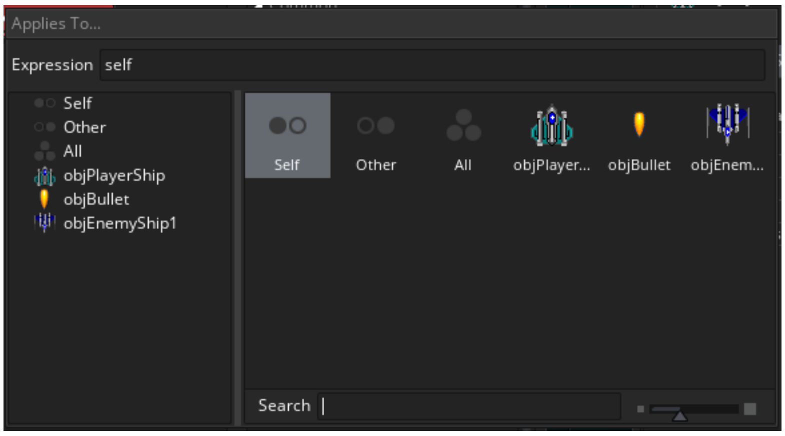
Create an object for this sprite, and call it something like "objEnemyShip1".

Place an enemy ship somewhere in your room, where you'll be able to shoot at it (near the top of the screen).

Now that you have your enemy ship object, set up an event for a collision between the enemy ship and your ship's bullet. Something like this…



Now comes the stuff that makes the pictures turn into a game. When the bullet collides with the enemy ship, you do not want to simply have the ship disappear. You do, however, want to make the bullet vanish. Now, instead of creating another event for the bullet when it collides with the enemy ship, you can use the Destroy Instance action () so drag that action into the actions window. At the moment the Destroy Instance action will destroy the enemy ship object but we want to destroy the bullet instead. Click on the little arrow and it will pop up a window to let you choose another instance to apply this action to.



Which option do you think you need to choose? If you think it is the 'objBullet' object you are close but not quite correct. If you choose objBullet as the option then it will destroy all bullet objects in the room which is kind of cool but you only want to destroy the bullet object for this collision so choose the Other option (Remember that we are programming from the point of view of the Enemy Ship – because this event is handled by the Enemy Ship.)

If you want, try your game out, and shoot your opponent. The bullet should disappear. That's exactly what needs to happen, but now to deal with the destroying of the enemy ship.

Make a new sprite using the file "Explosion.gif". You will see that the sprite is made of a series of slightly different images. It is an animated gif file. You will need to remove the background colour for each frame of the animation. Also set the Speed value to 20 frames per second. You can change the number later if you wish, bigger number will make the animation go faster.

Set its x origin to centre bottom just like the enemy ship and give it a suitable name. You don't need to set it to Precise collision checking because it won't be colliding with anything.

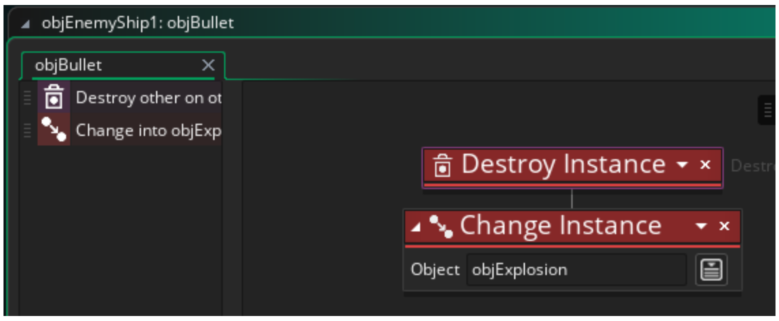
Create an object with that sprite. Call it 'objExplosion'.

Now that you have the explosion set up for, here's a cool way of destroying the ship.

Open the object preferences for the enemy ship.

Go to the collision event between the ship and the bullet.
After the destroy action, add in the 'Change instance' action () from the Instances group.

Click the button next to the Object textbox and select 'objExplosion'. You should see this:


objEnemyShip1: objBullet

objBullet

Destroy other on ot
Change into objExp

Destroy Instance

Change Instance

Object objExplosion

As you can see, this action changes one object instance into another. What you're going to do is: when the enemy ship is hit by the bullet, you'll change it into an explosion! Simple as that. The animation will play, and then, with one more event, it'll disappear.

If you run your game now, and shoot the enemy, the object will change into the right explosion, but you'll notice that it will keep playing the animation over and over again. Try it! You only want to do the explosion once, so hopefully there is an event for that situation.

Open up the object properties for your explosion object.
Add the event "Animation End" from the "Other" menu.

This event will trigger when the animation for the object has ended and is about to start again. Just add a destroy instance action, make sure it's destroying itself, and you're all done! Your enemy ship is now almost complete.

It should now be possible to shoot the enemy ship. Test your game.

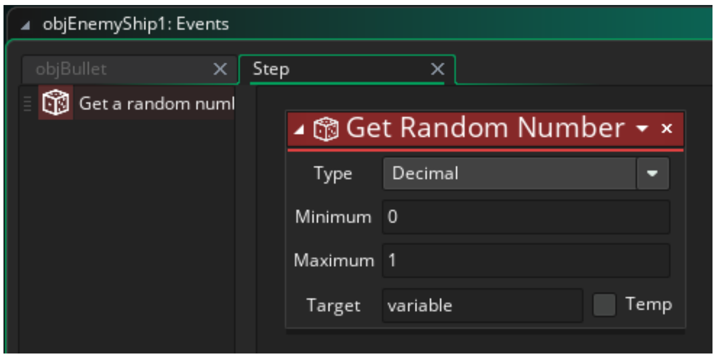
<u>**Letting the enemy fire back**</u>

Now comes another important part. Making the enemy ship fire. First thing that needs to be done is the creation of the enemy's bullet.

Make a new sprite called "sprEnemyBullet1" and load up the "Enemy Bullet 1.gif". Remove the background colour and set it to Precise Collision Mask. For this bullet, you need to set the origin at 16, 16 (Middle Centre), otherwise you're going to have your enemy bullets appear in weird positions. Make an object with this sprite, setting it up the same way that you set up the bullet for your ship, but make it so it moves down with a speed of 2. Make sure you put in the event to check if it's gone outside the room and if so destroy it.

Now, there are a number of ways of getting the enemy ship to fire, but the easiest one (requiring 2 actions) is to do it at random times. You'll notice in the screenshot of the control tab earlier in this tutorial, there was a dice conditional. This is a random thing that pretty much says "Do the following action when a random number between 1 and x is = to x". If you do this each step, and set the random x value to 2 * the number of steps per second, you'll get a ship that fires once every two seconds or so.

Open up the object editor for the enemy ship, and make a "Step" event.
In the action window for that event, add the Generate Random Number action () from the Random group in the toolbox, and you should get this screen.



Type: Decimal
Minimum: 0
Maximum: 1
Target: variable
Temp

Currently the type of number generated is set to Decimal which are numbers with decimal places (45.66). We only want to generate a whole number so change the Type setting to 'Integer'. Set the maximum to 10 and set the Target to 'varNumber' and tick the Temp option to store the random number in a temporary variable.

Now drag in the If Variable action from the Common group and change it to test if the varNumber variable is equal to 5. Then drag in and attach a Create Instance action to the If Variable action and make it create a new enemy bullet object relative to the enemy ship.

Run and test your game. Remember, there are 30 steps per second, so this event happens 30 times a second. It has a 10% chance of the If Variable action being true, 30 times per second which is way to fast. Try setting the maximum for the random number to 200. Experiment with other values and choose a value which creates bullets reasonably slowly.

Now, put in an action to create an Enemy Bullet object action after the random conditional, remembering to position it 'relative to' the ship and don't forget to put in the Start and End Block actions. Try it out.

Illustrate instantiation by creating multiple bad guys to shoot

Finally, we come to part 4 of this little part of the tutorial. Making multiple instances of the enemy fighters. Would you believe me if I just told you to place another enemy ship somewhere else in the room, and you're done? Well, that's all you have to do. When you create an instant of an object that automatically gives that object its own copy of all the information that you programmed into it, as well as the information that comes from the game. Place a row of enemy pieces into your room.

Now that you have a small army of enemy forces that you can valiantly defeat, it needs to be a bit more challenging because, at the moment, they can't kill you. That isn't very interesting, now is it? Time to put all the skills that this module has taught you to the test.

See if you can complete the following tasks.

Set up the collision between your ship and the enemy bullet 1.
Create an object for the explosion of your ship – either copy the current sprite you have for your explosion, or load/create a new one.
Position your explosion over your ship when it gets hit so that it looks realistic.
When the animation has ended for your ship exploding, then restart the level.

To restart the current room, you need this from the Rooms group in the toolbox.

Save your work.

Session 4: REVIEW PAGE **Name:** MIN SOE HTUT

Questions: Complete the following questions and be prepared to demonstrate techniques to your demonstrator. (You can then have your work verified.)

1. Why did you use the 'global' mouse event for firing?

   it will be one event for all of the room (all the level of the game)

2. Describe the 'conditional' event that handles only shooting one bullet at a time.

   Global left press
   if the count of the bullet is 0 then create the bullet

3. Describe the sequence of things that happen when a bullet hits an enemy ship.

   bullet hit the enemy ship it get deleted from the bullet view point

   bullet hits the enemy ship the ship deleted and replace with the exploration animation from the enemy ship viewpoint.

Verification: To assess your competency of this material your demonstrator will verify that you have completed the above questions, and can do the following:

1) Create a Diagonal Enemy Bullet object using the existing sprite. It should move diagonally left or right down the screen and should be destroyed if it goes outside the room.

Games – Clicky to Space Invaders – Session Five

Parenting: Different kinds of opponent

At this point you have a functional space invaders game. You can create waves of these enemy ships to destroy, but that'd get a bit boring to the eye. Perhaps another type of ship to blow up? The next section introduces the idea of *parenting,* not essential for game development, but helpful in making games more quickly, especially adding new objects that are similar to existing objects. The rough content of this session is:

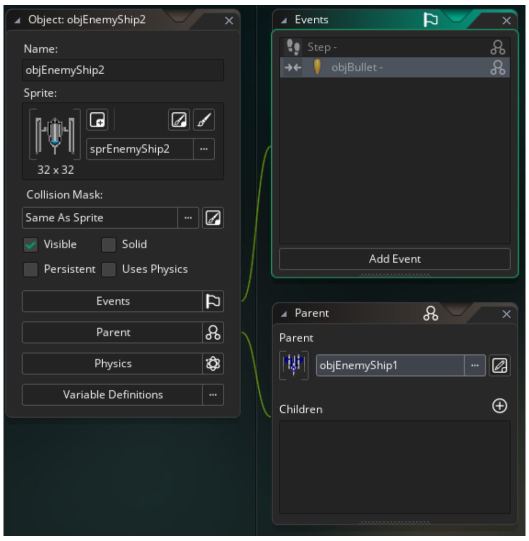
- Create another kind of object for bad guys, and illustrate parenting.
- Give one type of enemy object the ability to shoot at random
- Adding a win condition using the object counter.
- Extras.

<u>Create another kind of object for bad guys, and illustrate parenting</u>

Load up a new sprite called "sprEnemyShip2" using the Enemy Ship 2 image file, making sure you set your origin point the same way you did with the first enemy ship and also removing the background colour and setting collision mask to Precise, and create an object with that sprite. Give it a name, but don't do anything in the way of events yet. There's a feature in Game Maker that will make creating this ship a lot easier. It's called inheritance.

Inheritance is something that happens between parents and children. People and animals usually take on the physical traits of their parents. Things like skin, hair, eye colour and so on. The same thing can happen in computers, but in a different way. Game Maker has a "parenting" feature that allows one object to behave as though it was another, inheriting the traits of its parent while still having its own identity and image. The following experiment should make the idea clear.

For your new Enemy Ship 2 object, open the object properties window, and on the lower half of the window, find a field called "Parent". Use the dropdown list to its right to change this field to your first enemy ship (objEnemyShip1). You should end up with something looking like this.


Object: objEnemyShip2
Name: objEnemyShip2
Sprite: sprEnemyShip2
32 x 32
Collision Mask: Same As Sprite
Visible ✓ Solid
Persistent Uses Physics
Events
Step -
objBullet -
Parent
Parent: objEnemyShip1
Children

Notice that the Events window changes to show the events from enemy ship 1. You can select those events and see what is inside them but you can't make any changes. If you change the events in enemy ship 1 then those changes will also happen in Enemy Ship 2.

Now, in your room, place your new enemy ship somewhere where you can observe it, and run your game.

If you kept it running long enough, you would have noticed that your new enemy ship was firing even though you hadn't told it directly to do anything! You can also blow it up like any other ship. What has happened is your new ship has inherited or taken on the traits (events and actions) of its parent. It behaves just like whatever parent you set in that field. So, with only a small effort you have created a second type of enemy ship. It has a different appearance (sprite), but otherwise is exactly like the first type.

Try setting the Player's Ship to be the parent of Enemy Ship 2. You'll notice the ship snap to put its origin at the y co-ordinate of 400, just like your ship does, and when it gets hit by an enemy bullet, turns into an explosion (although off center because of the origin settings), and sets up to restart the level. Don't forget to set the Enemy Ship 2 parent back to enemy ship 1 when you have finished experimenting.

Give one type of enemy object the ability to shoot at random

Now, to add a little twist. Although your new enemy ship has taken on all the events and actions of the first enemy ship, it does not mean you can't create new ones. You can over-ride the inherited event handling or add responses to new events by simply adding the event in the child object. What that means is that even though your second enemy ship is mostly doing exactly the same thing as the first enemy ship, if you create a new event in that second ship, you'll over-ride what it's supposed to do (which is act like the first ship) for that event. So you can make this second ship fire twice as fast, or fire a different weapon, or move, but still have the collision events from the first enemy ship. We can demonstrate this by giving the new ship a different bullet.

Create a new sprite called sprEnemyBullet2 and load up the image "Enemy Bullet Blue.gif". Make sure that the origin is set appropriately, background colour is removed and set the collision mask to Precise. This sprite will be a new kind of enemy bullet. Create the object for the new enemy bullet and set the sprite.

Give the new bullet an event on creation to start moving in randomly chosen one of three directions down the screen with a speed of 2.

Give the new bullet an 'Outside room' event and make the bullet destroy itself if it goes off screen.

Create the "step" event for your second enemy ship and you will see that it replaces the Step event from the first enemy ship, and make it fire the new blue bullet after a certain amount of time.

As you can see, with a little bit of code, you've created a completely new type of enemy. In this way you can build yourself up a fleet to battle against, without having to reprogram the parts which all enemy have in common: blowing up when you shoot them!

Now that you're able to kill your enemy, and you have plenty of them to fight against, what happens when all the bad guys die? Well, for the moment, you can start the game again, but if you wanted to, you could build another room and make it harder.

Adding a win condition using the object counter

How to do the check to see if it's time to finish a game (or the current level)? Ask what the goal of this game/level is. In the case of Space Invaders, the goal is to destroy all the enemy ships. In Game Maker it is possible to check to see how many of a particular object are in a room at any one time – in our case to see if the number of enemy ships left is equal to zero. If there are none, advance to another room, quit or restart. To do this check, you use the Get Instance Count action and check to see if the number of instances of Enemy Ship 1 is = to 0. Note that as enemy ship 2 is a child of enemy ship 1, it's also included in the count, another great feature of inheritance, otherwise you'd have to check that there were none of either ship left.

An interesting question is then, which object should "do" the check. It doesn't really matter, provided the object is on screen at the time. One possibility is to have the player ship do the check in its step event.

Try adding a check to the player ship step event. A possible action is to restart. Another possibility is to move on to a new level. In Game Maker terms *levels* are *rooms*. So, to move to a new level, you must create a new room. Just make it, choose a background (or use the same one again), and fill it with enemy ships. Don't forget to put one player ship in as well. The new room can be made different to the first by changing the background, the layout or number of ships etc. Make the action on discovering that the level is finished, go to *next room*. That will be sufficient to get the new level started.

Extras

All that's left now is to add the frills. A score would be nice. Use the instructions given previously on creating your own scoring system and implement it for this game. There is one tricky issue you need to be careful of. Make sure that the check that the game is over doesn't slip in and stop the game just before your last kill is counted!

As it is currently set up the game does not check for bullets hitting other bullets. Some extra collision detection might improve play.

An *end screen* might be better than a little message box. Just make a room at the end whose backdrop has "Congratulations" in big letters, and a button to restart.

A BOSS! Lots of shooting games end with a giant enemy craft that is difficult to destroy.

Congratulations. You've made yourself a basic space invaders game! Admittedly it's not something that you're going to put up for sale, but it's a definite start. You've also learnt many of the basic ideas of game making: sprites and collisions, events and actions, objects and instantiation, to name a few.

To complete the module, choose an extra and see if you can add it to the game.

<u>Some final thoughts</u>

You may have been suspicious that whenever the trial game needed something, just the correct event or action turned out to be available. Did this happen because the example was very carefully chosen, or has Mark Overmars done a really brilliant job of working out just what functions are needed in games and making them available. We'll leave it to you to think about this, and ask what you think at the end of the theme.

Session 5: REVIEW PAGE **Name:** MIN SOE HTUT

Questions: Complete the following questions and be prepared to demonstrate techniques to your demonstrator. (You can then have your work verified.)

1. What object should have the event that detects winning?

   Count of enemies ship = 0

2. You may find that the game stops suddenly. It is normal in shooting games to delay for a little while after the last enemy is destroyed, to let remaining bullets finish their flights and for explosions to end. Why do you think this is done? How could you do it? (Outline a method, you don't need to implement it.)

   add a countdown or put the after the animation end then add event for to reset game or move to next room

3. Testing movement is easy enough. However it is quite difficult to test the end game – you have to play to completion to see if it works. What could you do to the game to make testing easier?

   have few enemies or room just to test the ending is working.

4. You may have noticed that player and enemy bullets just pass through each other. How could you fix this?

   Drage the bullet obj into the ship event and add destroy instance and change instance.

5. What extra did you implement.

   More Room and Ships

Verification: To assess your competency of this material your demonstrator will verify that you have completed the above questions, and can do the following:

1) Add another enemy ship to the game (load a new sprite), parented to the first enemy ship.
2) Make this new ship fire the diagonal bullet you created for verification in the previous session.