THE TECHNOLOGY AND CRAFT OF COMPUTER GAME DESIGN

An introductory course in computer game design

TUTORIALS, GRAPHICS, AND COURSEWARE BY:

MR. FRANCIS KNOBLAUCH

TECHNOLOGY EDUCATION TEACHER
CONWAY MIDDLE SCHOOL
ORLANDO, FLORIDA

GAME BUILDING TUTORIAL FOR THE GAME

G A L A C T I C M A I L

***BASED ON THE GAME CREATED BY:**

JACOB HABGOOD & MARK OVERMARS

***INCLUSIVE OF GAMEPLAY, GRAPHICAL/SOUND ASSETS, AND PROGRAMMING CONCEPTS**

GRAPHIC ASSET ILLUSTRATIONS BY:

KEV CROSSLEY

TUTORIAL AND COURSEWARE DOCUMENTS INCLUDE:

- ***Galactic Mail: Stages 1, 2, and 3***
- ***Galactic Mail: Stages 4, 5, and 6***

STUDENT RECORD DOCUMENTS INCLUDE:

- **Tutorial Guide: *Galactic Mail: Stages 1, 2, and 3***
- **Tutorial Guide: *Galactic Mail: Stages 4, 5, and 6***

COMPANION MATERIALS INCLUDE:

- **Glossary**
- **Concepts Explained**

REQUIRED SOFTWARE OR GAME ENGINE:

- **Game Maker 8.1 or Game Maker Studio**

REQUIRED DIGITAL ASSETS:

- ***Galactic Mail Assets**

This game build follows a procedure in STAGES similar to the one completed with *Evil Clutches. Galactic Mail* introduces new properties, as well as events and actions, that you have not seen before. There will be hypothesis and evaluation statements to complete on your tutorial guide, so don't look for much information for the object of the game just yet. In *Evil Clutches*, you were introduced to **coordinate systems** to locate objects and instances. In this game build, you further apply *x and y coordinates*. You will establish an **origin** so you can locate specific coordinates within an object. You will also be introduced to **wrap**, and other properties for creating basic **animation**, or motion, that enhances the game presentation. This simple animation method requires using an **image** and **sub-images,** and then creating properties so that this animated motion can be **random** or **controlled**. Find the *new vocabulary* definitions in the Glossary and write them on you Tutorial Guide.

BE SURE TO PAUSE TO COMPLETE HYPOTHESIS AND EVALUATIONS WHEN PROMPTED.

DO NOT USE THE TEST BUTTON IN THIS ACTIVITY UNTIL YOU ARE PROMPTED.

START HERE: Set up Gamemaker in **advanced mode** then save and name your file ***initials_galactic***

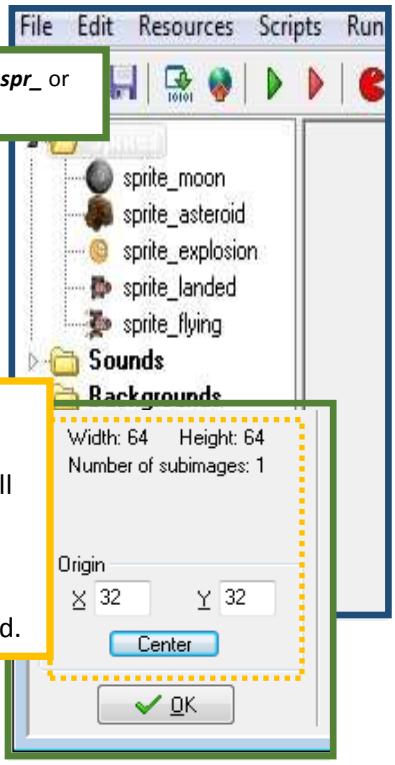
STAGE 1

Creating new sprite resources for the game

From **Resources** menu, choose **Create Sprite**.

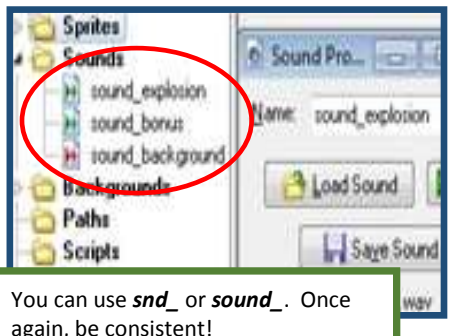
1- Click in the **Name** field and name it ***sprite_moon***.
2- Click **Load Sprite** choose ***Moon.png*** from the **Galactic Mail Assets** folder.
3- Click **Center** button to move the origin of the sprite to its middle.
NOTE: The width and height of this image is 64. By clicking Center, you will see the x and y values for the **origin** change to 32. This is half of the value of "64", hence "center".
4- Click **OK** to save and close.
5- Create sprites with a center **origin** for the ***asteroid*** and ***explosion*** using the assets ***Asteroid_strip180.png*** and ***Explosion_strip9.png***.
6- Create two sprites with centered **origins** called ***sprite_landed*** using ***Landed_strip72.png*** and another one called ***sprite_flying*** using ***Flying_strip72.png***.

Remember that you can use ***spr_*** or ***sprite_***. Just be consistent!



The **origin** is the point in a **coordinate system** where the counting for x and y begins. It is always identified as x = 0 and y = 0 on a grid. All of your **images** have an **origin** which can be moved. In this game, you will want all of your sprites to have a central **origin**. Read more about *coordinate systems* in Concepts Explained.

Sub-images: These sprite assets have 72 different small **sub-images** for creating an **animation** with 360 degree turning capability. Open one of these assets from your desktop assets folder and study it in a picture viewer. More information for **image** and **sub-image** can be found in Concepts Explained.



You can use ***snd_*** or ***sound_***. Once again, be consistent!

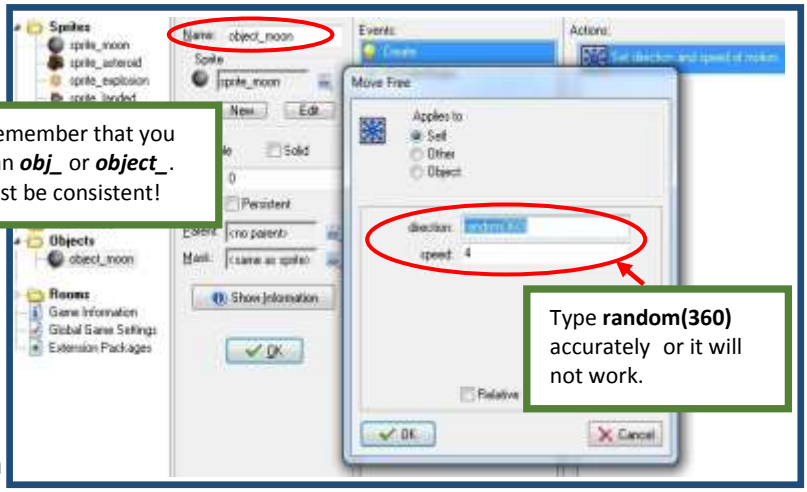
Creating new sound resources for the game

1- Select **Create Sound** from the **Resources** menu (the icon works too).
2- Name it ***sound_explosion*** and click **Load Sound**.
Choose **Explosion.wav** from the Assets folder. Click **OK** when done.
3- Create sounds for ***bonus*** and ***music*** using the ***bonus.wav*** and ***Music.mp3*** files.

GO TO THE FILE DROP DOWN AND SAVE OR CLICK THE DISKETTE IN THE TOOLBAR

Creating the moon object

1- From the **Resources** menu, choose **Create** object.
2- Name the ***object_moon*** and choose the **moon** sprite.

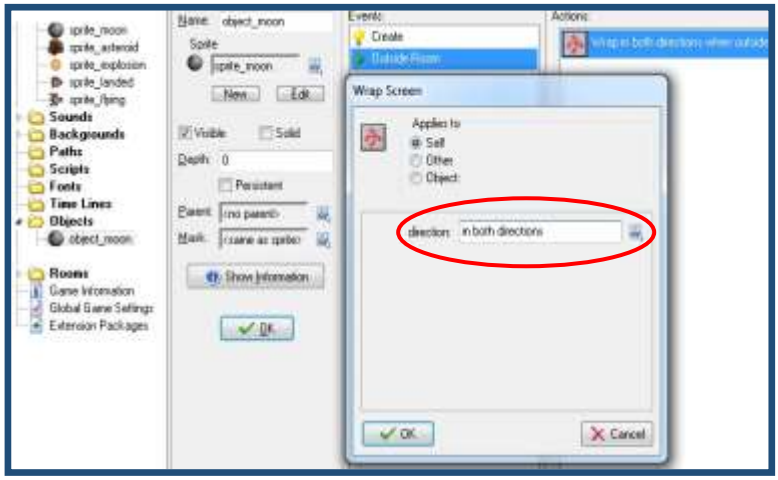


Adding a create event to the moon object

1- Click the **Add Event** button and choose **Create**.
2- Include the **Move Free** action in the Actions list.
3- Enter a **Speed** of **4** and type **random(360)** in the **Direction** property. This will make the moon move in random directions.

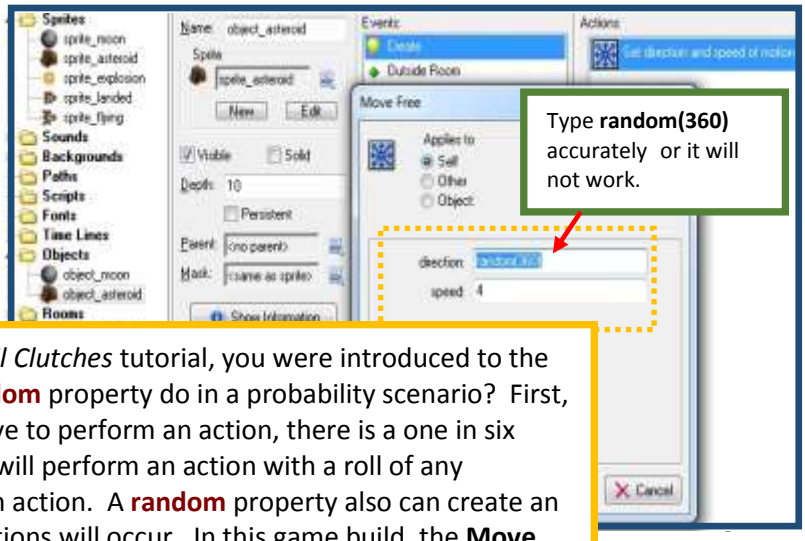
Including a Wrap action for the moon object

1- Click the **Add Event** button, choose **Other** and select **Outside Room** from the pop up menu. Drag and drop the **Wrap Screen** action for this event.
2- Select "**in both directions"** for the direction property.
3- Click **OK** to save and close these Properties.



Creating the Asteroid object

1- Create a new object called ***object_asteroid*** and giving it ***sprite_asteroid***.
2- Set the **Depth** to **10**.
3- Add the **Create Event** then drag the **Move Free** action. Type **random(360)** in the **Direction** field and a **Speed** of **4**.
4- Add the **Other**, **Outside Room** event and **Wrap Screen** action **in both directions**.



<u>Random in programming probability:</u> In the *Evil Clutches* tutorial, you were introduced to the concept of *probability*. What does creating a **random** property do in a probability scenario? First, think of a roll of one die. If I am waiting to roll a five to perform an action, there is a one in six chance of getting my five. On the other hand, if I will perform an action with a roll of any number, then there is a six out of six chance for an action. A **random** property also can create an equal possibility that any one of many possible actions will occur. In this game build, the **Move Free** property creates a 100% chance (or 1 to 1) that the asteroid will **Move Free** (in other words, it is going float around). Since there are 360 directional possibilities, **random** makes the chances the same for each possibility (1 in 360). Based on this property, the program selects any one of the 360 directions, and then the **Move Free** action occurs no matter which one it chooses. That's a lot of different directions, making the outcome unpredictable! You *can't expect* an asteroid to move in any one direction based on this probability, but you *can expect* that it is going somewhere. There is some *certainty* here. Wherever it goes, it will go at a **Speed of 4**. We will study *certainty* in a future tutorial. *Random*, *random-number generator*, *probability*, and *certainty* in programming can be further researched in Concepts Explained.

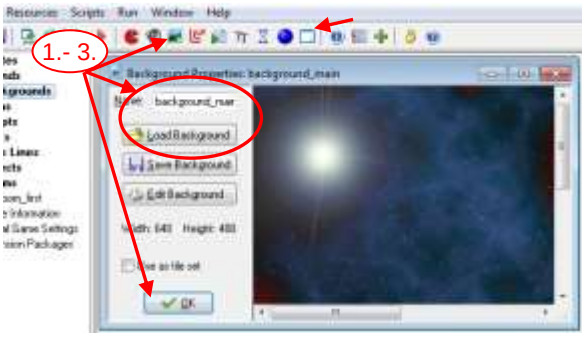
SAVE NOW

DO THIS IN YOUR TUTORIAL GUIDE

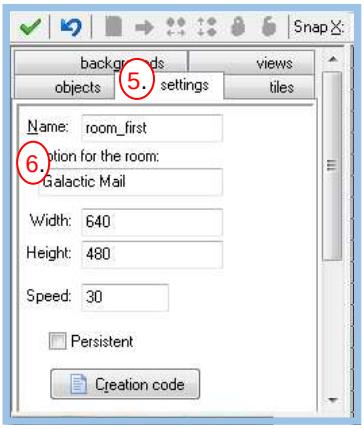
STAGE 1 HYPOTHESIS STATEMENTS: Now it's time to predict the behaviors of the sprite and object properties applied to the assets thus far.

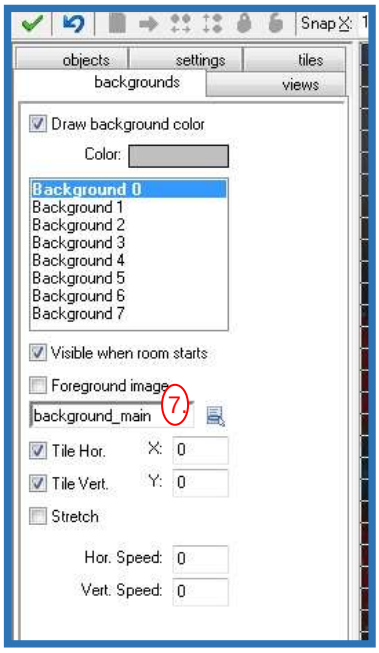
- **In a couple of sentences, explain the moon object behaviors that you expect to see in the game file.**

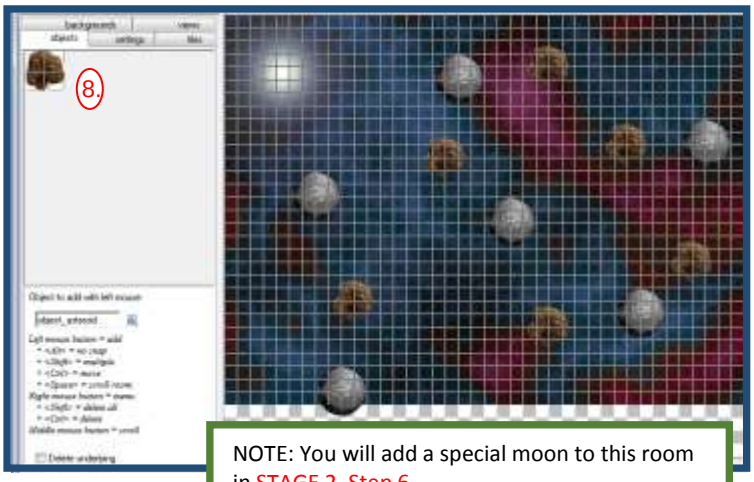
Creating a room with moon and asteroid instances



1- Select **Create Background** from the Resources menu. Name it ***background_main***.
2- Click the **Load Background** choosing ***Background.png***.
3- Click **OK** to save and close the properties.
4- Select **Create Room** from the Resources menu.
5- Select the **settings** tab and enter the name the ***room_first***.
6- Type ***Galactic Mail*** the **caption** field.
7- Select the **backgrounds** tab. Click the menu icon to change the <*no background*> field to "***background_main***".
8- Select the **Objects** tab and place some asteroids and moons (about five to seven of each) in the room.
9- Close the **Room Properties** form by clicking the green check mark.







NOTE: You will add a special moon to this room in STAGE 2, Step 6.

PRESS THE <ctrl> KEY AND PRESS THE <s> KEY. THIS IS JUST A "SHOW OFF" WAY TO SAVE!

At this point, go ahead and perform a test (▸). Check that the moons and asteroids are moving in **randomly**, each in different directions. Do they **wrap** by leaving the room only to appear on the other side of the screen? Do the asteroids pass behind the moons because of the *depth* properties? If not, recheck the steps for making them move.

DON'T SAVE….YOU ALREADY SAVED. BUT IF YOU DO, IT'S OKAY! CONTINUE ON NEXT PAGE

DO THIS IN YOUR TUTORIAL GUIDE

STAGE 1 EVALUATION: Now it's time to explain "why" your game properties are working or "why not". Be sure to indicate if your hypothesis is "valid" or "invalid"

Answer the following in a couple of sentences.

- **Does the action that you see and the control of the objects look like your hypothesis statements?**
- **If NOT, what was the difference in behaviors?**
- **What needed to be fixed and how did you fix it?**

STAGE 2:

Creating the special moon object

1- **Right click** the moon object and select **Duplicate**. A copy of the moon will appear in the list. This moon object will be an exact copy of the first moon object, including properties.

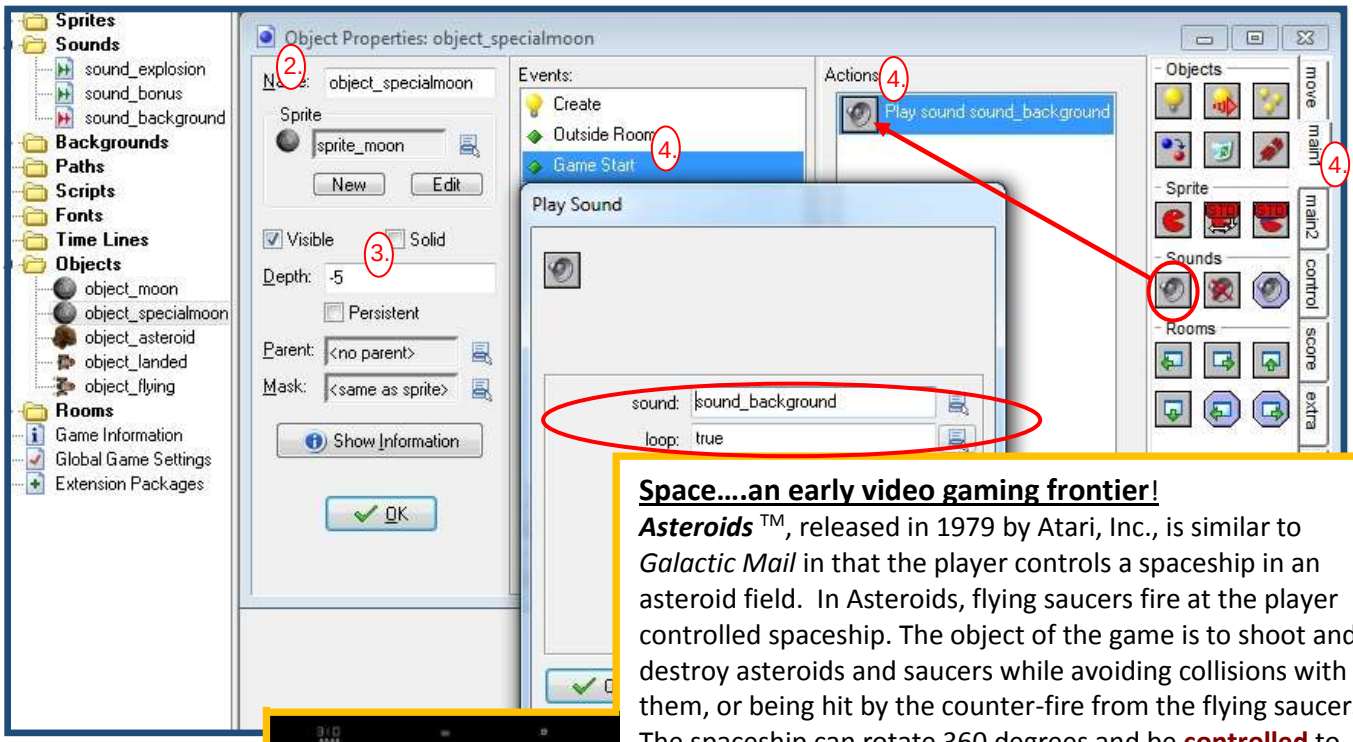
2- Double-click on that new item to open the object properties and change the name to ***object_specialmoon***. Type it exactly as shown, as the name will be recognized by another property in a different step.

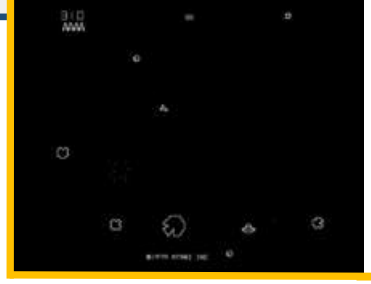
3- Set the **Depth** to **-5**. This will property always puts instances of this moon in front of all others.

4- Create an **Other**, **Game Start** event, then drag and drop a **Play Sound** action (**main 1** tab). Select the background music "***sound_background***" sound and set loop to **"true"**.

5- Click **OK** to in the *Play Sound* window then **OK** to save and close the properties.

6- Open the ***room_first*** and add one instance of a special moon (see graphic in STAGE 1b).





Space….an early video gaming frontier!

Asteroids ™, released in 1979 by Atari, Inc., is similar to *Galactic Mail* in that the player controls a spaceship in an asteroid field. In Asteroids, flying saucers fire at the player controlled spaceship. The object of the game is to shoot and destroy asteroids and saucers while avoiding collisions with them, or being hit by the counter-fire from the flying saucers. The spaceship can rotate 360 degrees and be **controlled** to fire in any direction. The asteroids float **randomly** and rotate 360 degrees and can **wrap** off and on the screen. The **controlled** object of the spaceship, along with the asteroids, have **animation** features similar to those in this game build. Read more on *random properties*, *wrapping*, and *animation* with Gamemaker in Concepts Explained.

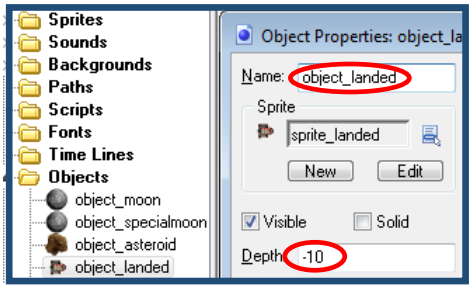
SAVE

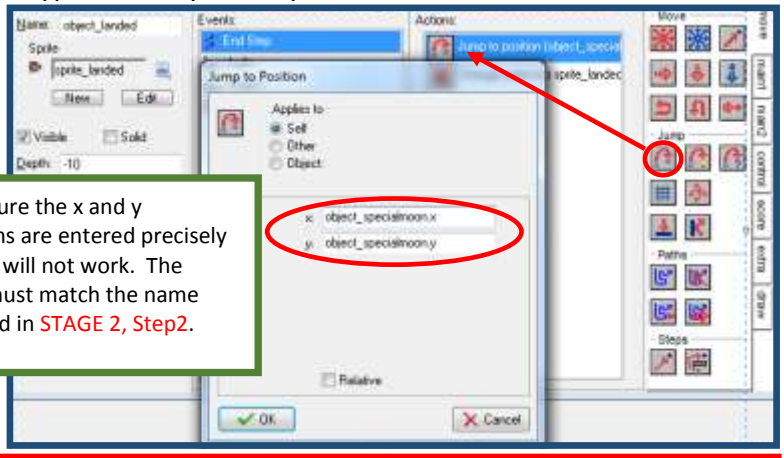
Creating the landed rocket object

1- Create a new object called ***object_landed*** and using ***sprite_landed*** sprite. Set the **Depth** to **-10**. NOTE: Do you want the rocket in front of or behind a moon when it lands? Write about this in the STAGE 2 HYPOTHESIS.

2- Add a **Step**, with **End Step** event from the pop-up menu.

3- Include a **Jump to Position** action. In the **X value**, type ***object_specialmoon.x*** and in the **Y value** type ***object_specialmoon.y*** . These names must be typed exactly or they will not work! Click **OK**.





Make sure the x and y positions are entered precisely or they will not work. The entry must match the name assigned in STAGE 2, Step2.

DO THIS ON YOUR TUTORIAL GUIDE

STAGE 2 HYPOTHESIS STATEMENTS: Now it's time to predict the behaviors of the rocket object properties applied to the assets.

- **In a couple of sentences, explain the two different rocket object behaviors that you expect to observe when testing the game file.**

PLACE AN INSTANCE OF THE ROCKET INTO THE ROOM.

(▸) GO AHEAD AND TEST YOUR HYPOTHESIS

The rocket should jump to a special moon. The **animation** of the rocket will continually spin around because of the multiple **sub-images** of the rocket rotating **randomly** with the moons. You are about to fix that! The music should be playing.

Control: Recall that the moon and asteroid each have a **Move Free** property set with **random(360)** allowing 360 different directional possibilities for 360° of free rotation and movement. A rocket, on the other hand, must be **controlled** by a skilled "postal employee" who can pilot the spaceship through this hazardous galaxy. This rocket must also have the capability to fly in all directions, but unlike the randomly floating moons and asteroids, it must be guided by **controls**. As a programmer, you must create properties to control the rocket. Naturally, there are 360 directional possibilities. The rocket image has 72 sub-images pointing in 72 different directions. So the rocket sprite must be able to rotate 360°, but in 72 different positions. Stated mathematically, **DIRECTION / 5 = # OF SUB-IMAGES or 360/5 = 72.** This is the first of several properties that will settle that rocket down so it can be **controlled** by the pilot. Consider this as you complete your STAGE 3 HYPOTHESIS. You can also refer to *Concepts Explained* at any time.

STAGE 3:

Adding a change sprite action

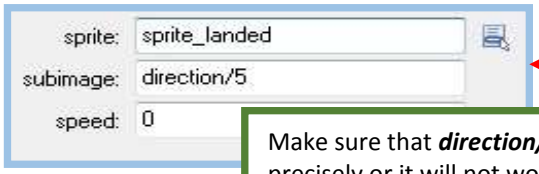
1- Open the object properties for ***object_landed***.

2- Reopen the **End Step** event, then add a **Change Sprite** action from the **main1 tab**.

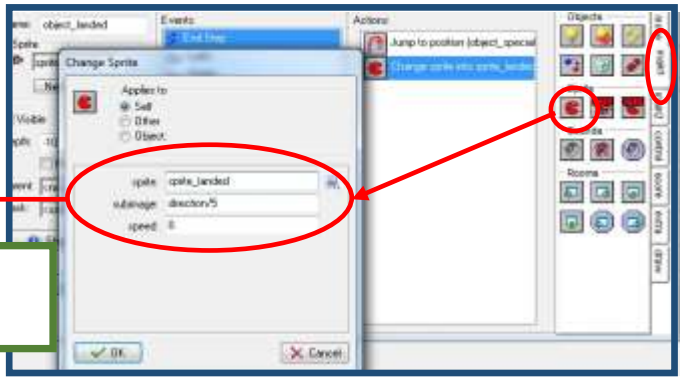
3- When the window opens, select the ***sprite_landed*** from the sprite menu.

4- Then type **direction/5** in **Subimage**.

5- Set **Speed** to **0** (preventing the sprite from self-animating).



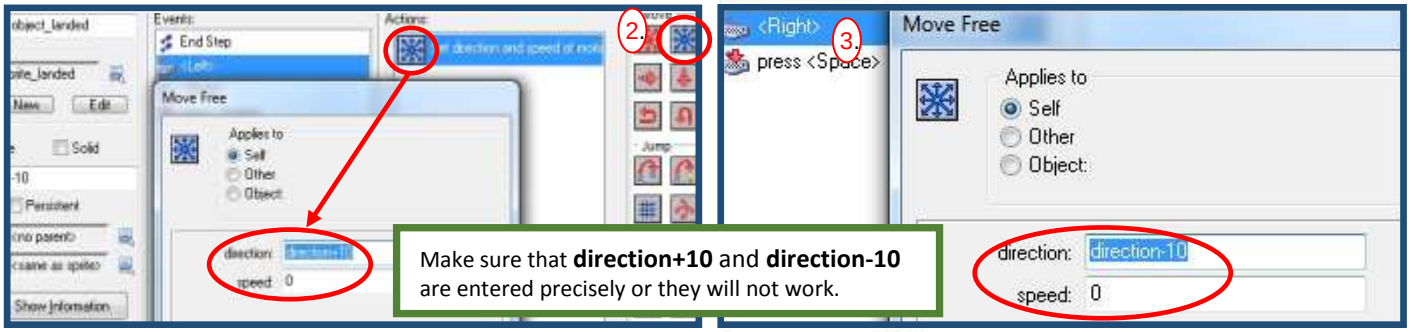
Make sure that ***direction/5*** is entered precisely or it will not work.



SAVE

Keyboard events and actions for the landed rocket object

1- Add a **keyboard event,** the select **<left>**.
2- Include a **Move Free** action. Set **Direction** property type **direction+10** then Set **Speed** to **0**.
3- Add another **Keyboard event** for the **<right>** key. Include a **Move Free** action. Set **Direction** property type **direction-10** then Set **Speed** to **0**.

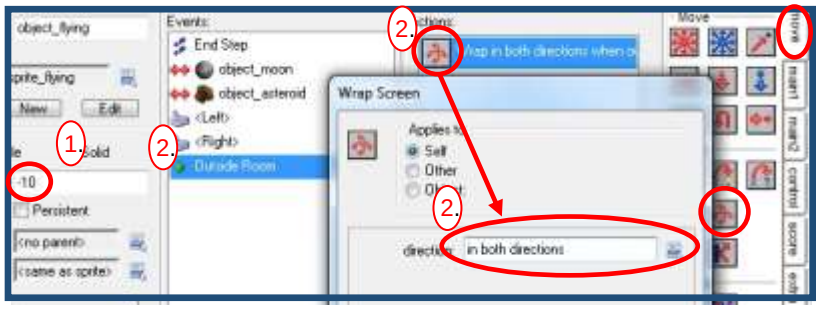


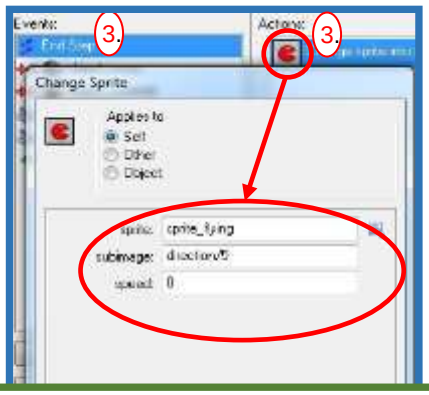
Enhancing the animation with a flying rocket object

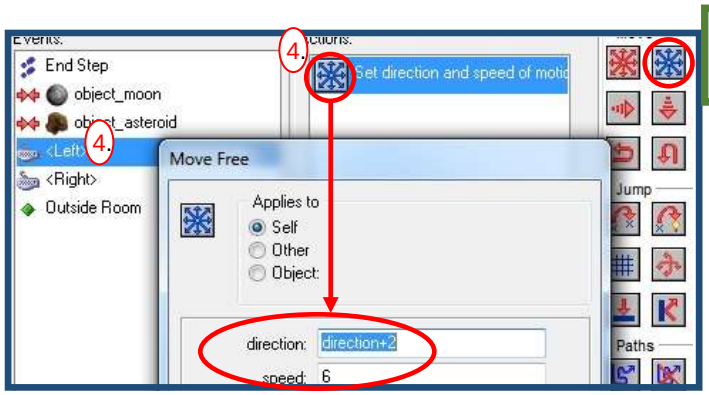
1- Create a an object called ***object_flying*** with the ***sprite_flying*** sprite. Set **Depth** to **-10**, keeping this object in font of special moons.
2- Add an **Other** event, selecting **Outside Room**, then add a **Wrap Screen** action. Select "**in both directions"** for the direction property. Click **OK**.
3- Add an **End Step** event. Add a **Change Sprite** action from the **main1** tab, with ***sprite_flying*** , type **direction/5** in **Subimage** field and set **Speed** to **0**.
NOTE: Do you remember why we do this? Write about this in your STAGE 3 HYPOTHESIS.
4- Add a **Keyboard event** for the **<left>** key. Include a **Move Free** action. Set **Direction** property type **direction+2** then Set **Speed** to **6**.
5- Add another **Keyboard event** for the **<right>** key. Include a **Move Free** action. Set **Direction** property type **direction-2** then Set **Speed** to **6**.
NOTE: Why are the speed and direction values different than the landed rocket? Write about this in your STAGE 3 HYPOTHESIS.

Controls……

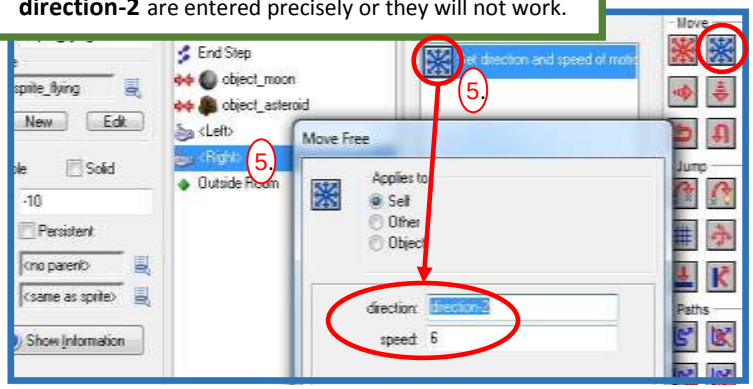

Earth Mail Galactic Mail







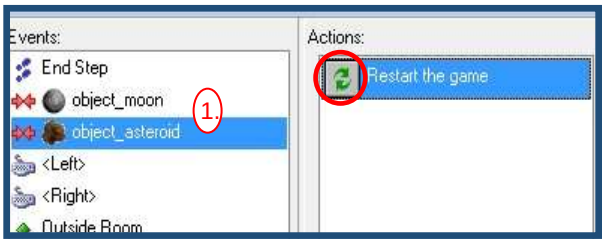
Make sure that **direction/5, direction+2** and **direction-2** are entered precisely or they will not work.

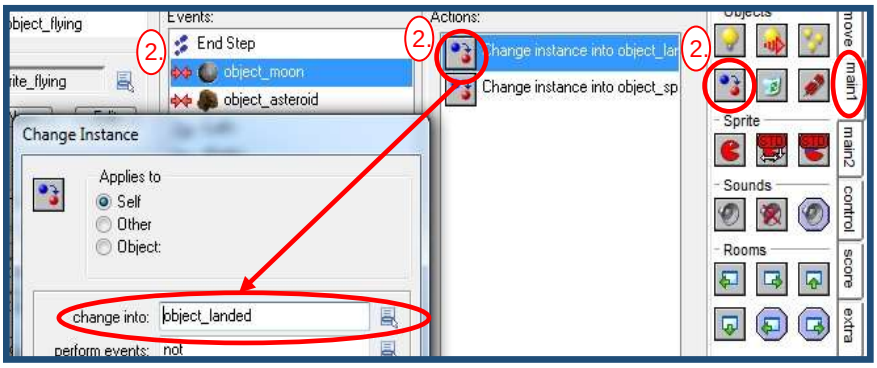


SAVE

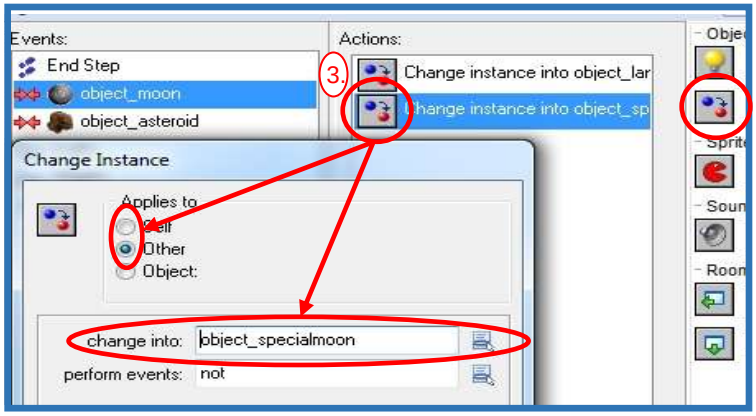
Adding Collision events to the flying rocket object

1- Add a **Collision** event with ***object_asteroid***, then drag **Restart Game** from **Main 2** to the actions list.
2- Add a **Collision** event with ***object_moon,*** then drag a **Change Instance** action from **main1**. Set the **change into** menu to ***object_landed***.
3- Drag another **Change Instance** action. Change the **Applies** bullet from **Self** to **Other.** Set the change into menu to ***object_specialmoon.***





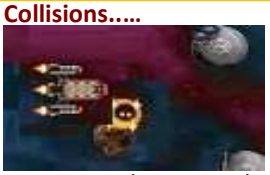
<u>More logic systems</u>: Recall from *Evil Clutches* that an event always causes something else to happen. Each of the programming properties shown is creating a *cause and effect*. When the properties effect each other, they form a *logic system*.



Collisions…..

Earth Mail

Collisions…..



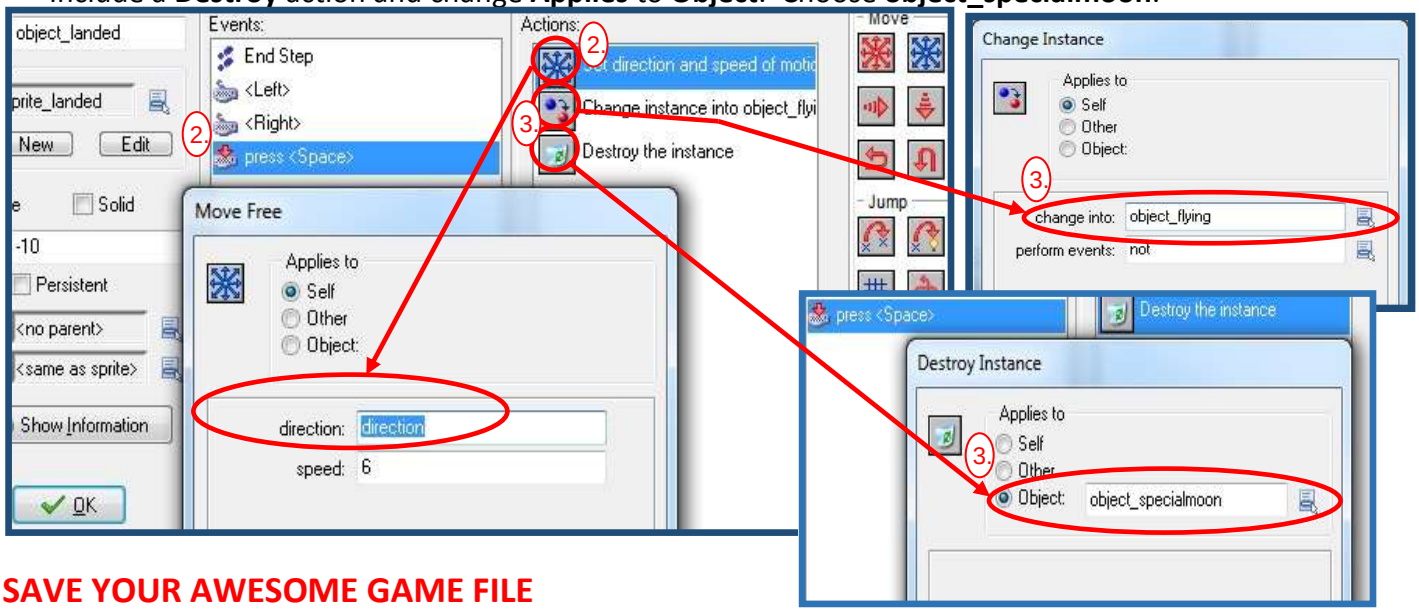
Galactic Mail

DO THIS IN THE SPACE PROVIDED ON YOUR TUTORIAL GUIDE

Review *conditional statements* in *Concepts Explained* or in the *Evil Clutches* tutorial. **Write a *conditional statement* (IF/THEN) for each of actions that occur as a result of collision events shown above.** You could have three different statements, or you can try combining the two actions from the moon collision into one statement, for a total of two.

Adding a key press event to the landed rocket object

1- Open the object properties for the ***object_landed***.
2- Add a **Key press**, **<space>** event with a **Move Free** action. Type **direction** in the **Direction** property and set **Speed** to **6**.
3- Add a **Change instance** and change object to ***object_flying***.
Include a **Destroy** action and change **Applies** to **Object**. Choose **object_specialmoon**.



SAVE YOUR AWESOME GAME FILE

DO THIS ON YOUR TUTORIAL GUIDE

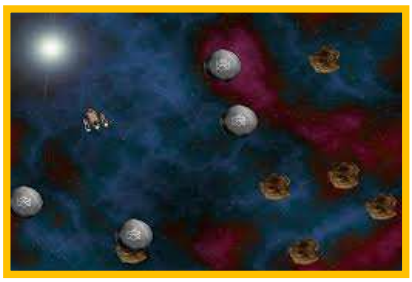
STAGE 3 HYPOTHESIS STATEMENTS: Now it is time to predict the way that all sprites and objects will behave in the room.

- **In a couple of sentences, explain the behaviors that you expect to observe. Be sure to include how you, as a player, refer to the concepts you applied in this stage.**

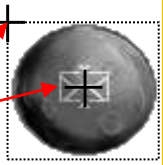
Now it's time to test your game, so go ahead and click on the green triangle (▶).

DOES THE ACTION THAT YOU SEE AND THE CONTROL OF THE OBJECTS MEET YOUR HYPOTHESIS STATEMENTS?

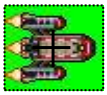
You should have moons and asteroids scattering **randomly** through the galaxy of your room, with the asteroids in a rotation. With the controls for the rockets in place, your rocket should be a **controlled** object. Collisions with asteroids force you to start over. Cool music makes the pilot relax.



Putting concepts together: To make more sense of your *logic system*, let's think about the cause and effect of your properties.



- When you created your sprite, you moved the **origins** from to the center of each sprite.
- On both rocket (flying and landed) you centered the origin and set the depth to -10 so it will always be in front of the moons (set at -5).
- You set properties to control or "fly" the rocket, including a property that "jumps" the rocket from moon to moon.



- You told the rocket to change from the flying sprite to landed when on a moon, centering the objects (origin to origin).

Conditional Statements for Controls: A LOGIC SYSTEM

IF left arrow key is pressed, THEN rocket turns left.
IF right arrow is pressed, THEN rocket turns right.
IF space bar is pressed, THEN rocket jumps off of moon.
IF rocket collides with moon, THEN rocket changes looks from flying to landed.
IF rocket collides with moon, THEN rocket appears in front of moon and centered.

Troubleshooting ("debugging")

If you see these "bugs", how will you "debug" them?


Rocket not centered


Rocket behind moon

"Neither snow nor rain nor heat nor gloom of night stays these couriers from the swift completion of their appointed rounds."

-U.S. Postal Service

DO THIS ON YOUR TUTORIAL GUIDE

STAGE 1, 2, 3 TEST & EVALUATION: Now it is time to explain "why" your game is working or "why not"….that's evaluation.

Answer the following in a couple of sentences.

- **Does the action that you see and the control of the objects look like your hypothesis statements?**
- **If NOT, what was the difference in behaviors?**
- **What needed to be fixed and how did you fix it?**

BE SURE THAT YOU HAVE SUCCESSFULLY COMPLETED STAGES 1, 2, AND 3 BEFORE PROCEDING TO THE STAGES 4, 5, AND 6 TUTORIAL. ALL PROPERTIES MUST BE WORKING BASED ON THE DESCRIPTION ABOVE.