Directions for Super Rainbow Reef Game

Directions from the Game Maker's Apprentice book by Jacob Habgood and Mark Overmars

Game Description

The monstrous Biglegs have driven the peace-loving creatures of Rainbow Reef from their ancestral home. Despite their inexperience in the ways of war, Pop and Katch have invented a way of combining their skills to fight back against the Biglegs. For this incredible feat, Pop must bounce from Katch's shell to attack the evil invaders. Katch must then move quickly to save Pop from plummeting into the deep waters below. The cowardly Biglegs often retreat behind coral defenses, so our heroes must be prepared to smash their way through if they are to finally drive the Biglegs from Rainbow Reef.

There will be no direct control over Pop's movement, and he'll bounce freely around a playing area enclosed by walls on all sides except the base. The left and right arrow keys will move Katch horizontally along the base in order to bounce Pop from Katch's shell and stop him from falling out of the level. The collision point along Katch's shell will determine the direction of Pop's bounce, and so allow the player to control his movement. Bounces toward the left will send Pop left and bounces toward the right will send him right. Pop's movement is also affected by gravity, and each time he collides with Katch, he gets slightly faster so that the game becomes increasingly difficult.

The game will have several levels, each containing a number of Biglegs that Pop must collide with in order to complete the level. Most levels will also contain coral block defenses, which must be knocked out of the way in order to reach the Biglegs. Breaking blocks will score extra points and special blocks give the player extra rewards, but they don't have to be destroyed to finish a level. If Pop leaves the screen, the player loses a life and Pop is brought back into play. Once three lives have been lost, the game ends and the high score table is displayed.

Features list:

Enemies

- Large stationary Biglegs
- Small stationary Biglegs
- Large Biglegs that move horizontally
- Small Biglegs that move horizontally

Blocks

- Multicolored blocks that can be destroyed for points
- Solid blocks that cannot be destroyed
- Blocks that must be hit twice before they are destroyed
- Invisible solid blocks that cannot be destroyed
- Blocks that create two extra copies of Pop when destroyed
- Blocks that give the player an extra life

Game Framework:
Create a standard framework of set components for any game you create.

- A title screen displaying the name of the game and buttons to start a new game, load a saved game, get help, display the high scores table and quit the game. This is referred to as the front end of a game. This screen also starts the background music and sets the initial score.
- The game
- The end screen that shows the game is over. It displays a congratulatory message and the high score table and then brings you to the front end again.

Creating the front-end resources for the game:

1. After launching Game Maker, start a new empty game.
2. Create a sprite called sprite_title using the Title.png from the folder.
3. Create 5 button sprites: sprite_button_start, sprite_button_load, sprite_button_help, sprite_button_scores, and sprite_button_quit and assign the appropriate sprites.
4. Create a new background called background1 using Background1.png for the folder.
5. Create a new sound resource called sound_music using Music.mp3 from the folder.

Create the title object resource for the front end:

1. Create a new object called object_title using the title sprite.
2. Add a **Create** event and include a **Set Score** to set the **Score** to 0.
3. Add an **Other**, **Game Start** even and include a **Play Sound** action (main1). Choose the background music and set **Loop** to **true** so the music continually plays.
4. Click **OK** to close the form.

Creating the button objects resources for the front end:

1. Create a new object called object_buttonstart and use the start button sprite.
2. Add a **Mouse**, **Left pressed** event and include the **Next Room** action (**main1**). This will start when the user clicks on the instance's screen position with the left mouse button.
3. Click **OK** to close the Object properties form.
4. Create a new object called object_button_load and assign the load button sprite.
5. Add a **Mouse**, **Left pressed** event and include the **Load Game** action (**main2**). File Name must be the same name as the file so we use the save the game to later on. Leave it as the default setting..
6. Click **OK** to close the Object properties form.
7. Create a new button called object_button_help and assign to the help button sprite.
8. Add a **Mouse**, **Left pressed** event and include the **Show Info** action (**main2**). This shows the player the text entered under the **Game Information**.
9. Click **OK** to close the Object properties form.
10. Create a fourth button object called object_button_scores and assign the score button sprite.
11. Add a **Mouse**, **Left pressed** event and include the **Show Highscore** action (**score**). Make the settings the way you want so the table looks nice.
12. Click **OK** to close the Object properties form.
13. Create a final object called object_button_quit and assign the quit button sprite.
14. Add the **Mouse**, **Left Pressed** event and include the **End Game** action (**main2**).
15. Click **OK** to close the Object properties form.

Create the front end room resource for the game:

1. Create a new room resource called room_frontend (**settings** tab). Give the room a caption and scale the room window so you can see as much of the room as possible.
2. Switch to the backgrounds tab. Click the menu icon to the right where it says <no background> and click on the background from the pop up menu.
3. Next switch to the objects tab. Place one instance of the button objects along the bottom of the room. A logical order would be Start, Load, Help, Scores and Quit.
4. Select the title object and position it to the center of the room (you can move an instance by holding the CTRL key.

Adding game information to the game:

1. Double click on **Game Information** near the bottom of the resource list.
2. Type the name of the game, your name as creator, a description of the objectives and a description of the controls.
3. Click on the green checkmark at the top left to close the editor.

We will next create the completion screen. The process is similar to the start screen. It will display text congratulating the player and adds 1000 points to the player's score. There will be a short pause and the high score table will be displayed. Then the player returns to the opening screen.
To achieve this, we are going to use new events and actions called alarms. These work like a countdown for triggering alarm events. These actions are given a set amount of time to wait, and then it executes when the time runs out. Each instance of an object can have up to 13 alarms.

Creating a congratulations object that uses alarm events and actions:

1. Create a new sprite called sprite_contgratulations using congratulations.png.
2. Create a new object called object_congratulations using the congratulations sprite.
3. Add a **Create** event and include the **Set Alarm** action (**main2**). Set the **Number of Steps** to 120 to create a delay of 4 seconds (30 steps per second). Leave In **Alarm No** set to **Alarm 0**.
4. Include a **Set Score** action below the alarm using a **New Score** of 1000 and enable **Relative** so the score is added on.
5. Add an **Alarm**, **Alarm 0** event. This happens four seconds after the alarm action is executed. Include a **Show HighScore** action and give it the same background and font settings as before.
6. Include the **Different Room** action (**main1**) and select front end room.
7. Click **OK** to close the properties form.

Creating a Completion room for the game:

1. Create a new room called room_completed (settings tab) and give it an appropriate caption.
2. Switch to the background tab. Click the menu icon to the right of <no background> and select the background from the popup.
3. Switch to the objects tab. Select the congrats object and position it to the top-left corner of the room.

Test your work. Try the buttons. It should take you to the end of the game since there are no levels yet. You can also put your name on the high score table. **Save** the game as rainbow1.gmk

Creating new sprite resources for Pop, Katch and the wall:

1. Create a sprite called sprite_wall using wall.png from the folder.
2. Create a sprite called sprite_pop using Pop_strip45.png. Click the Center button to place the origin at its center.
3. Create a sprite called sprite_katch using Katch_strip24.png. Click o the Center button again.

Creating the wall object resource for the game:

1. Create a new object called object_wall and select the wall sprite.
2. Enable the **Solid** option and click **OK** to close the form.

Creating the Katch object resource for the game:

1. Create a new object called object_katch and assign Katch's sprite.
2. Add a **Keyboard**, **<left>** event.
3. Include the **Check Object** conditional action (**control**). Select the wall object, set **X** to -10 and **Y** to 0 and enable **Relative**. This will check the wall object 10 pixels to the left of Katch. Enable the **NOT** option. This reverses the condition so the next action (moving Katch) will be executed if there is not a wall object in the way.
4. Include a **Jump to Position** action (**move**). Set the same values for **X** and **Y** (-10 and 0) and enable **Relative** so Katch moves into position that we checked was free of walls.
5. Add a **Keyboard**, **<right>** event.
6. Include a **Check Object** action (**control**). Select the wall object, set **X** to 10 and **Y** to 0, enable **Relative**. Enable the **NOT** option. This performs the same check for walls to the right.
7. Include the **Jump to Position** action (**move**). Set **X** to 10 and **Y** to 0 and enable **Relative**. This moves Katch to the right if there are no walls blocking.

Creating the Pop object resource for the game:

1. Create a new object called object_pop and assign the pop sprite. Set its Depth to 10 so that it appears behind other objects in the game.
2. Add a **Create** event and include a **Move Free** action with a **Speed** of 12. We want Pop to start moving upward but we will add a variation by typing random(60)+60 in **Direction**. The random(60) will produce a random number between 0 and 60 and then adding 60 to it brings the range to 60 and 120. This is the angle that the bounce will work from.
3. Include a **Set Gravity** action (**move**) setting **Direction** to 270 (meaning downward) and **Gravity** to 0.2. This is the downward speed that will be added to Pop's current speed for each step pulling him slowly downward.
4. Add a **Collision** event with the wall object and include a **Bounce** action. Set the **Precise** option to **Precisely** so it takes into account the exact appearance of the colliding sprites to calculate the result of the collision.

Adding a collision event to the Pop object for colliding with the Katch object:

1. Add a **Collision** event with the Katch object and include a **Move Free** action. Type 90+object_katch.x-x in **Direction** and speed+0.3 in **Speed** (adding 0.3 to the current speed).
2. We need a restart the room when Pop falls off the bottom of the screen. Add the **Other**, **Outside Room** event and include a **Restart Room** action (**main1**).

Creating a new test room resource for the game:

1. Right click the room_completed in the resource and choose Insert Room for the popup. This will put the room before room_completed and open the properties window for the new room.
2. Switch to setting tab and call the room room_test.
3. Switch to the backgrounds tab and set the background for the room.
4. The wall sprites are 20x20 pixels so set the **Snap X** and **Y** to 20 at the top of the Room properties form.
5. Switch to the objects tab. Select the wall object and place wall instances all the way along the left, top and right edge of the screen. Hold the SHIFT key to add multiple instances. Right click to delete those that you put in the wrong place.
6. Select the Katch object and place one instance in the middle of the bottom of the screen. Add an instance of Pop somewhere in the middle of the room.

SAVE and test you work. Save as rainbow2.gmk

Creating the Bigleg object resources for the game:

1. Create a sprite called sprite_bigleg and use Bigleg_strip24.png.
2. Create an object called object_bigleg and assign the bigleg sprite.
3. Add a **Collision** event with Pop and include a **Destroy Instance** action (**main1**).
4. Include a **Set Score** action (**score**) with **New Score** set to 200 and **Relative** enabled (Relative will add the value to the current score).

Creating a controller object resource for the game:

1. Create an object called object_controller. No sprite is assigned.
2. Add the **Step**, **Step event** and include a **Test Instance Count** action (**control**). Indicate the Biglegs object as the object to count and leave the default values to check when the number of Biglegs =0.
3. Include a **Start Block** action.
4. Include a **Sleep action** (**main2**) with **Milliseconds** set to 1000 (1 second).
5. Include a **Next Room** action (**main1**)
6. Finish the block with an **End Block**.

Duplicating test room resources for the game:

1. Reopen the test room.
2. Put one instance of the controller object in the bottom left corner.
3. Put 2 instances of the Bigleg object in the top left and right corners of the room.
4. Close the properties form for the room.
5. Right click the test room and choose Duplicate from the pop up.
6. Switch to the setting and give the room a new name and new caption.
7. Switch to the objects tab and right click the Biglegs instances to remove them.

8. Add a few instances of the Biglegs object in the top center of the room. Add some instances of the wall object below them to make it harder for Pop to reach them. It should look like an upside down T from the top and then a line below the center bar.

SAVE your work and test the game to make sure things are working properly.

Creating the small Biglegs object resources:

1. Create a sprite called sprite_bigleg_small using Bigleg_small_stirp24.png.
2. Create an object called object_bigleg_small using small Bigleg sprite.
3. Click on the menu icon next to the **Parent** field and select object_bigleg as the parent.
4. Click **OK** to close the form.

The small Biglegs will now do everything the large object does. That is due to the fact that the large Biglegs is acting as a parent. It therefore inherits the behaviors of the large Biglegs. When the small Bigleg collides with Pop, the player's score will increase by 200 and the Bigleg will get destroyed. The controller object will count all Biglegs the same.
Add some small Biglegs into the test rooms. Make sure they get destroyed when Pop hits them.

Creating the moving Bigleg object resource:

1. Create an object called object_bigleg_move and use the normal Bigleg sprite.
2. Click the menu icon next to parent and choose object_bigleg as parent. Set the Depth to 10 so it appears behind other objects.
3. Add a **Create** even and include a **Move Fixed** action. Select both left and right directions and set the **Speed** to 8. Movement will be random.
4. Add a **Collision** event with the wall and include a **Reverse Horizontal** action. Click **OK** to close the form.

Creating the small moving Bigleg object resource:

1. Create an object called object_bigleg_move_small and assign the small Bigleg sprite.
2. Click the menu icon next to Parent and choose object_bigleg_move. Set the **Depth** to 10 so it appears behind other objects.
3. Click **OK** to close the form.

Add a few moving Biglegs and small moving Biglegs to the test levels. Make sure the game is working properly. Save this as rainbow3.png.

Using Parent properties saves so much time. If you want to make changes after creating objects, changes made to the parent will change all objects using that object.

Parent Rules

Inheriting events: A child inherits all events (actions) of the parent. A child can have its own events as well but these do not apply to the parent. When both child and parent have the same event with different actions, the child actions are for the child and the parent ones are used by the parent.
Actions on Objects: When an action refers to a parent object, this includes instances of the child too. However, when an action refers to the child, it doesn't affect the parent.
Collisions with Objects: A collision event with the parent object also applies to collisions with children of that object. However, a collision event with a child object does not apply to collisions with parent objects.

Parenting Objects: Parents can have parents. You cannot create cycles of parents, so if P is the parent of C, C cannot be the parent of P. You cannot be your own father or grandfather.

Setting the lives in the create event of the title object:

1. Double click the title object in the resource list.
2. Select the **Create** event to view the Actions list. Include a **Set Lives** action (**score**) with a value of 3.

Editing the Pop object to add an event for being outside of the room:

1. Double click the Pop object from the resource list.
2. Select the **Outside Room** event and remove **Restart Room** action (left click once and hit delete key. Include a **Destroy Instance**.
3. Include the **Test Instance Count** (**control**). Indicate Pop object as object to count and leave default values to see if this is last Pop leaving the screen.
4. Include the **Start Block**.
5. Include a **Set Lives** with **New Lives** set to -1 and **Relative** enabled. This will subtract a life.
6. Include a **Create Instance**. Select Pop set **X** to 320 and **Y** to 300.
7. Include an **End Block** and Click **OK** to close properties.

Editing the controller object to add an event for having no more lives:

1. Double click the controller object.
2. Add the **Other**, **No More Lives** event and include a **Show Highscore** action. Use the same background and font as initially set.
3. Include the **Different Room** action and indicate the front end room. Click **OK** to close properties form.

Creating a sprite for the controller object to draw as lives:

1. Create a sprite called sprite_katch_small using Katch_small.png.
2. Double click on controller object in resources list.
3. Add a **Draw event** and include **Draw Life Images** action (**score**). Set **X** to 25 and **Y** to 470 and select small Katch sprite. Click **OK** to close the properties form.

Save your work and play the game. Save as rainbow4.gmk. You should see three small images of Katch in the bottom left of the screen. One should disappear when Pop goes off the screen and the game ends with the high score table when all are gone.

Normal Blocks:
Now we are going to create a level that has blocks of different colors to break.
We will use parents to create a number of different colored blocks. The blocks will all behave the same and the colors just there to make it look pretty.

Creating block object resources for the game:

1. Create seven sprites called sprite_block1 to sprite_block7 using block1.png to block7.png.
2. Create a new object called object_block1 and assign block1 sprite. Enable **Solid** and close the form.
3. Create 6 more objects for the other six blocks using the appropriate sprites. Enable **Solid** for each and set Parent to object-block1. Now the other blocks are the children.

Editing the Pop object to add a collision event with the blocks:

1. Reopen the Pop's object properties form.
2. Add a **Collision** event with object_block1 and include the **Bounce** action with **Precise** set to **Precisely**.
3. Include a **Set Score** action with **New Score** set to 20 and **Relative** enabled.
4. Include a **Destroy Instance** action and set Applies to **Other**. This means the block, not Pop is destroyed.

Solid Blocks:
These are blocks that Pop cannot destroy. They will behave the same as the wall object so we will make it the parent.

Creating a solid block resource for the game:

1. Create a sprite called sprite_block_solid using block_solid1.png
2. Create a new object called object_block_solid and assign the solid block as a sprite. Check **Solid** and give it the wall object as a **Parent**. Click **OK** to close the properties form.

Creating an invisible block resource for the game:

1. Create a new object called object_block_solid_inv and assign the solid block as a sprite. Enable **Solid** and give it the wall object as a **Parent**. Also disable **Visible**. Click **OK** to close the properties form.

Creating a double block sprite and object resources:

1. Create a sprite called sprite_block_double using Block_double.png.
2. Create a new object called object_block_double. Use the double sprite and enable **Solid**. Click **OK** to close the properties form.

Adding a collision event to the Pop object for colliding with double blocks:

1. Reopen Pop's object properties form and add a Collision event with the double block object.
2. Include the **Bounce** action in this event and set **Precise** to **Precisely**.
3. Also include a **Set Score** action with a **New Score** of 20 and **Relative** enabled.
4. Include a **Change Instance** action. Set **Applies** to **Other** and select object_block1 so the double block changes into a normal block.
5. Click **OK** to close the properties form.

Creating the split block sprite and object resources:

1. Create a sprite called sprite_block_split using block_split.png.
2. Create a new object called object_block_split and assign to the new sprite. Enable **Solid** and set the Parent to object_block1.
3. Add a **Collision** event with the Pop object and include a **Create Instance**. Set object to object_Pop then type other.x into **X** and other.y in **Y**.
4. Include another **Create** Instance the same as step 3.

Creating a life block sprite and object resources:

1. Create a sprite called sprite_block_life using block_life.png.
2. Create a new object called object_block_life. Enable **Solid** and set the Parent to object_block1.

3. Add a **Collision** with Pop object and include a **Set Lives**. Set **New Lives** to 1 and enable the **Relative** option.

Add these new blocks to the test level. **Save** your work and test it. Name your file rainbow5.gmk.

Creating sound resources and playing them in the appropriate events:

1. Create sounds using the files sound_wall.wav, sound_block_wav, sound_katch.wav, sound_bigleg.wav, sound_lost.wav, sound_click.wav. and give them the appropriate names.
2. **Play** the “wall sound” in the Pop object’s Collision event with the wall object.
3. **Play** the “block sound” in the Pop object’s Collision event with the block object and double block object.
4. **Play** the “katch sound” in the Pop object’s Collision event with the Katch object.
5. **Play** the “lost sound” in the Pop object’s Outside Room event.
6. **Play** the ‘bigleg sound” in the Bigleg object’s Collision event with the Pop object.
7. **Play** the “click sound” at the top of the Left Pressed event of each of the button objects.

Adding a save game action to the controller object:

1. Reopen the controller object’s properties form.
2. Add a **key Press**, **Letters**, **E** event and include the **Save Game** action (**main2**). Leave File Name set to the default as it is the same as we used for the Load Game action in the front-end.
3. Include a **Display Message** (**main2**) and type “Game Saved” in Message so that the user knows the game was saved.
4. Add a line to the **Game Information** describing how the game can be saved.

When the ESC key is pressed, the game ends completely. We will change this so it brings the player to the front end.

Editing the controller object and global game settings to disable the ESC key:

1. Reopen the controller object’s properties form.
2. Add a **Key Press**, **Others**, **<Escape>** event and include the **Different Room** action (**main1**). Select Front end room and click **OK** to close the properties form.
3. Double click the **Global Game Settings** at the bottom of the resources list.
4. Switch to the **Other** tab and disable **Let <Esc>** end game option.
5. Also **disable** the **Let <F5>** save the game and **<F6>** load game option.
6. Click **OK** to close the form.

Creating a new stationary Pop object resource:

1. Create a new object called object_pop_start. Assign Pop sprite and a Depth of 10.
2. Add a **create** event and include a **Set Alarm** (**main2**) with 30 steps (1 second).
3. Add an **Alarm**, **Alarm 0** event and include **Change Instance** action. Set **Change Into** to object_pop and indicate **Yes** to **Perform Events**. New Pop object starts moving n a random direction
4. Reopen the Pop object’s properties form and select the **Outside Room** event. Double click **Create** Instance and change the Pop object to object_pop_start.
5. Go through each room replacing the Pop objects with the start Pop object.

Editing the controller object to add cheats:

1. Reopen the controller's properties form.
2. Add a **Key Press, Letters, N** and include **Next Room** action.
3. Add a **Key Press, Letters, P** and include **Previous Room** action.
4. Add a **Key Press, Letters, R** and include **Restart Room** action.
5. Add a **Key Press, Letters, L** and include **Set Lives**. Set **New Lives** to 1 and enable **Relative**.
6. Create levels that containing walls, the controller, stationary Pop, and Katch. You can copy them and add Bliglegs and blocks where you want them.
7. **Save** your work as **rainbowcomplete.gmk**