

Game Maker

Chapter 6

By Marc Long

Rainbow Reef

Game Description

“The monstrous Biglegs have driven the peace-loving creatures of Rainbow Reef from their ancestral homes. 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!”

Columns Score: 200



Movement Defined

*“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.”*

Challenges And Rewards

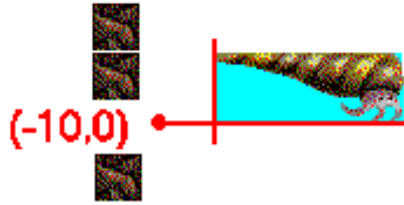
*“The game will have several **levels**, each containing a number of Biglegs that Pop must collide with in order to complete the level. Most levels will also contain coral block **defenses**, which must be knocked out of the way in order to reach the Biglegs. Breaking blocks will score extra **points** 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 a **high-score table** is displayed.”*

Creating a Front End

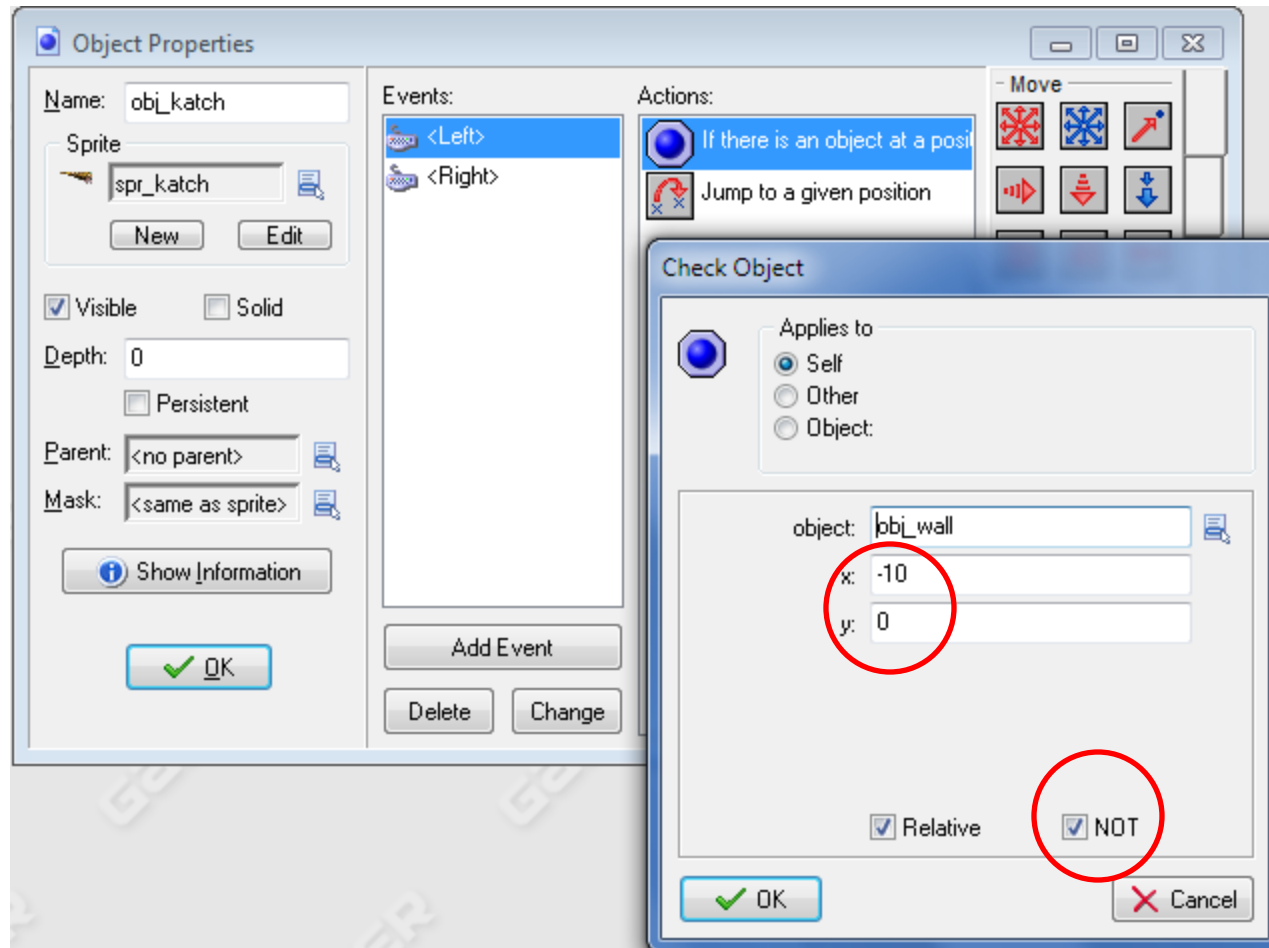
- The *front-end* to a game refers to the part of the game that allows the player to make selections.



Collision with the Wall



- Katch will only move left if there is no object on his left.
- The coordinates are to Katch's left.
- The NOT says its okay to move to the coordinates just as long as there is not an object is in the way.



NOT

- NOT reverses the truth of the condition being tested.

Object Properties

Name:

Sprite:

Visible Solid

Depth:

Persistent

Parent:

Mask:

Events:

- <Left>
- <Right>

Actions:

- If there is an object at a position
- Jump to a given position

Information about: obj_katch

Information about object: obj_katch

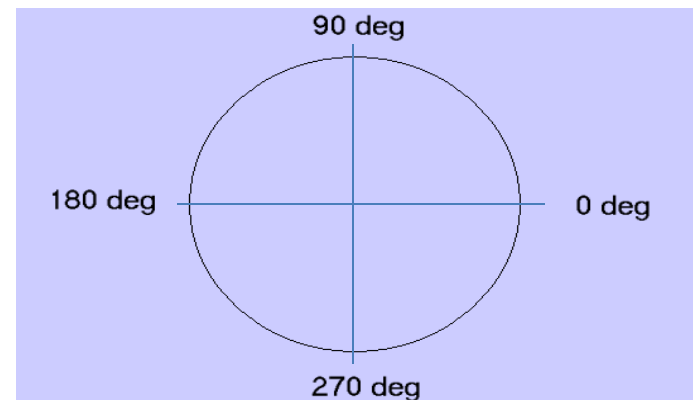
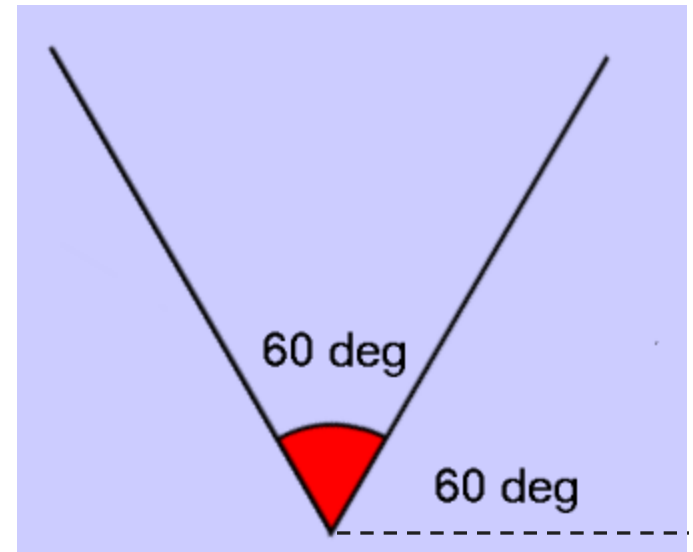
Sprite: spr_katch
Solid: false
Visible: true
Depth: 0
Persistent: false
Parent: <no parent>
Mask: <same as sprite>

Keyboard Event for <Left> Key:
if at relative position (-10,0) there is not object obj_wall
move relative to position (-10,0)

Keyboard Event for <Right> Key:
if at relative position (10,0) there is not object obj_wall
move relative to position (10,0)

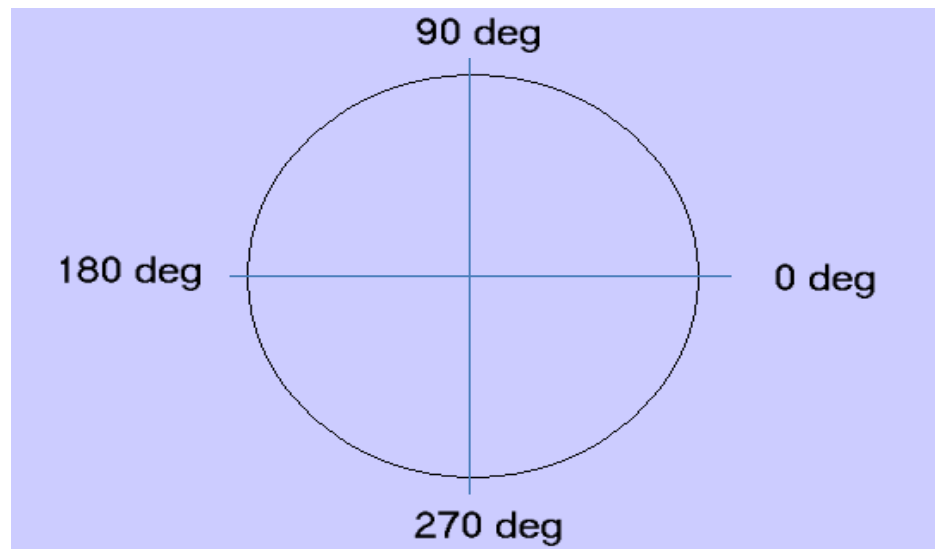
Random

- How does `random(60) + 60` work?
- `Random (60)` will generate a random number between 0 and 59.
- The random value is then added to a minimum value of 60.



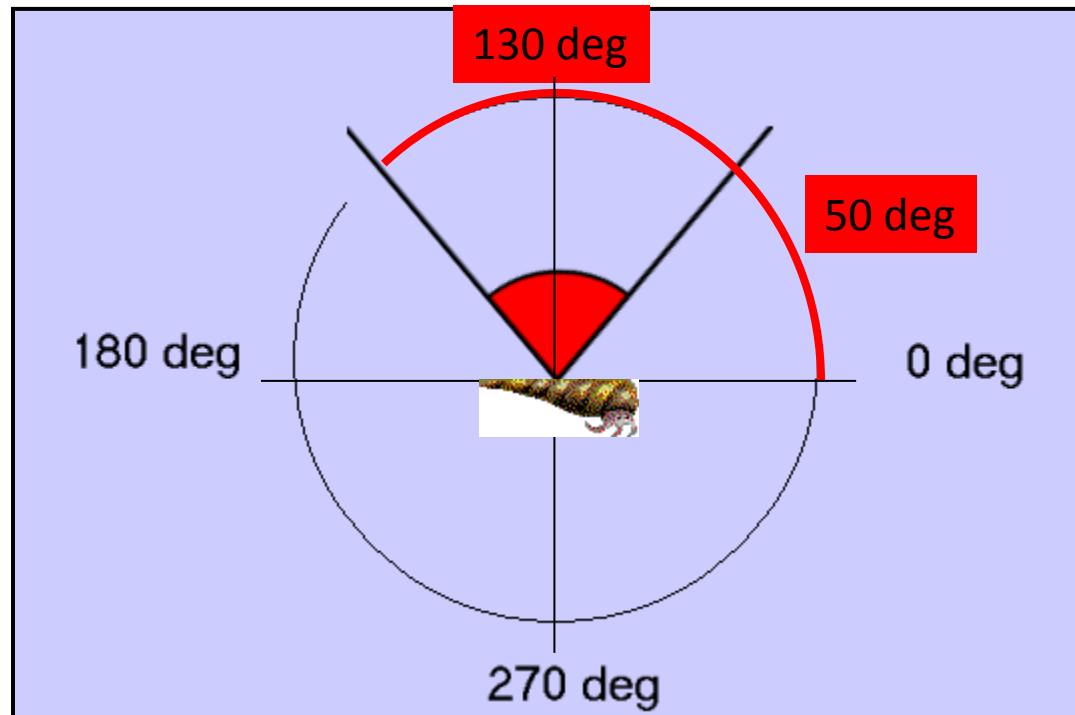
Gravity

- You can apply gravity to an object by setting Direction to 270 (directly downward), and Gravity to 0.2.



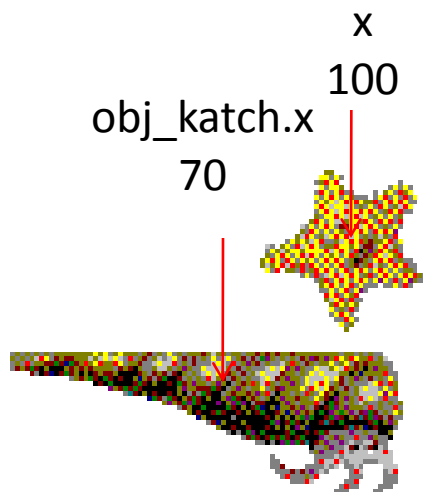
Controlling the Bounce

- The user will be able to control the angle of bounce based upon the location of the collision with Katch's shell.
- It will bounce between 50 and 130 degrees.



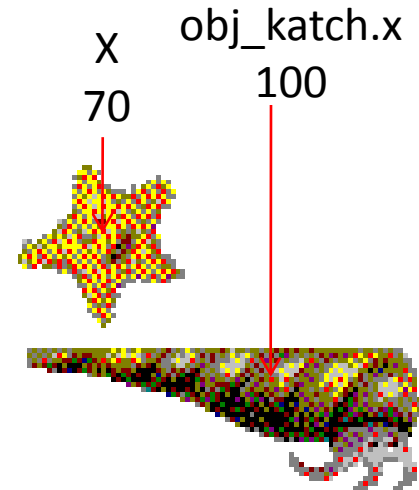
Computing the Bounce

- We can get an object's own x-position using the x variable, and the x-position of Katch using obj_katch.x
- If Pop's x-position is larger than Katch's x-position, then he has landed on the right.
- The difference between Pop's and Katch's horizontal positions by subtracting one from the other: obj_katch.x-x.



$$\text{obj_katch.x} - x = 70 - 100 = -30$$

We need a direction smaller than 90 degrees,
 $\text{obj_katch.x} - x + 90 = 70 - 100 + 90 = 60$




$$\text{obj_katch.x} - x = 100 - 70 = 30$$

We need a direction larger than 90 degrees,
 $\text{obj_katch.x} - x + 90 = 100 - 70 + 90 = 120$

Advancing to the Next Room

- The object controller keeps track of the number of BigLegs left in the room. When they are gone, the user has completed the level.

Name:

Sprite
 






Visible Solid

Depth:

Events:

- Step

Actions:

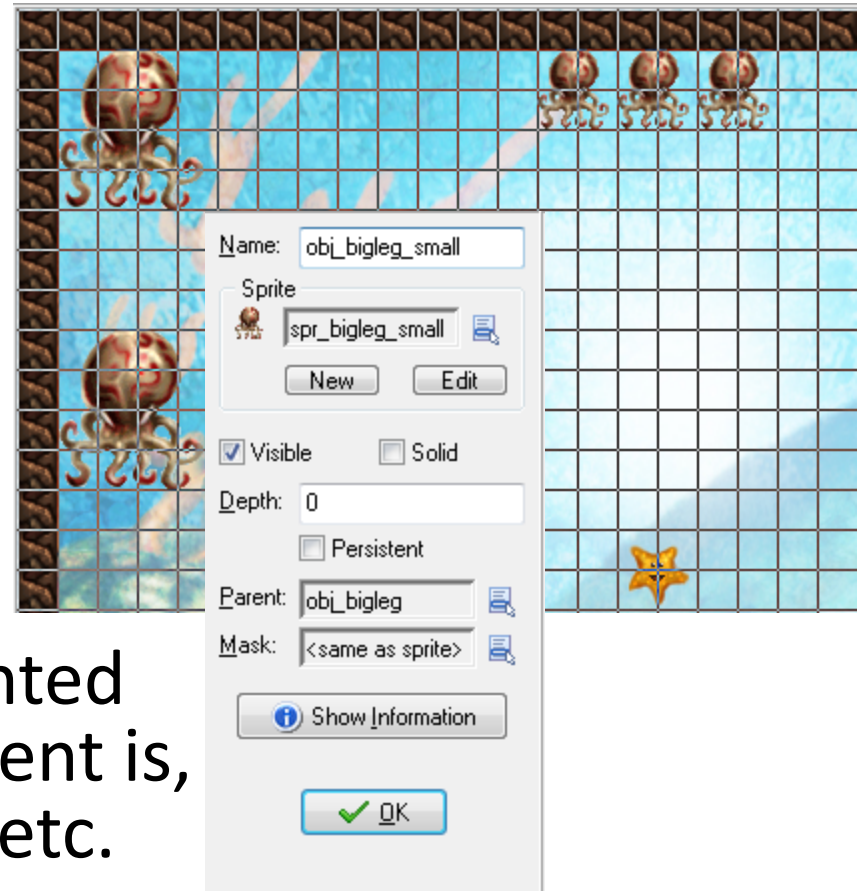
-  123 If the number of instances is a value
 -  Start of a block
 -  Sleep 1000 milliseconds
 -  Go to next room
 -  End of a block

Adding a Level of Complexity

- It would be more challenging to make a smaller Bigleg, making it more difficult to hit.
- To make this, we could repeat all the earlier steps and let the controller object count both the normal objects and the small objects.
- However, there is a much quicker and easier way to do this: by using *parents*.

Parent/Child

- By setting the parent field to “obj_bigleg” we are saying “obj_bigleg_small” inherits all of the behaviors (events and actions) of its parent obj_bigleg.
- Small BigLegs will be counted in the scores just as its parent is, and destroyed the same, etc.

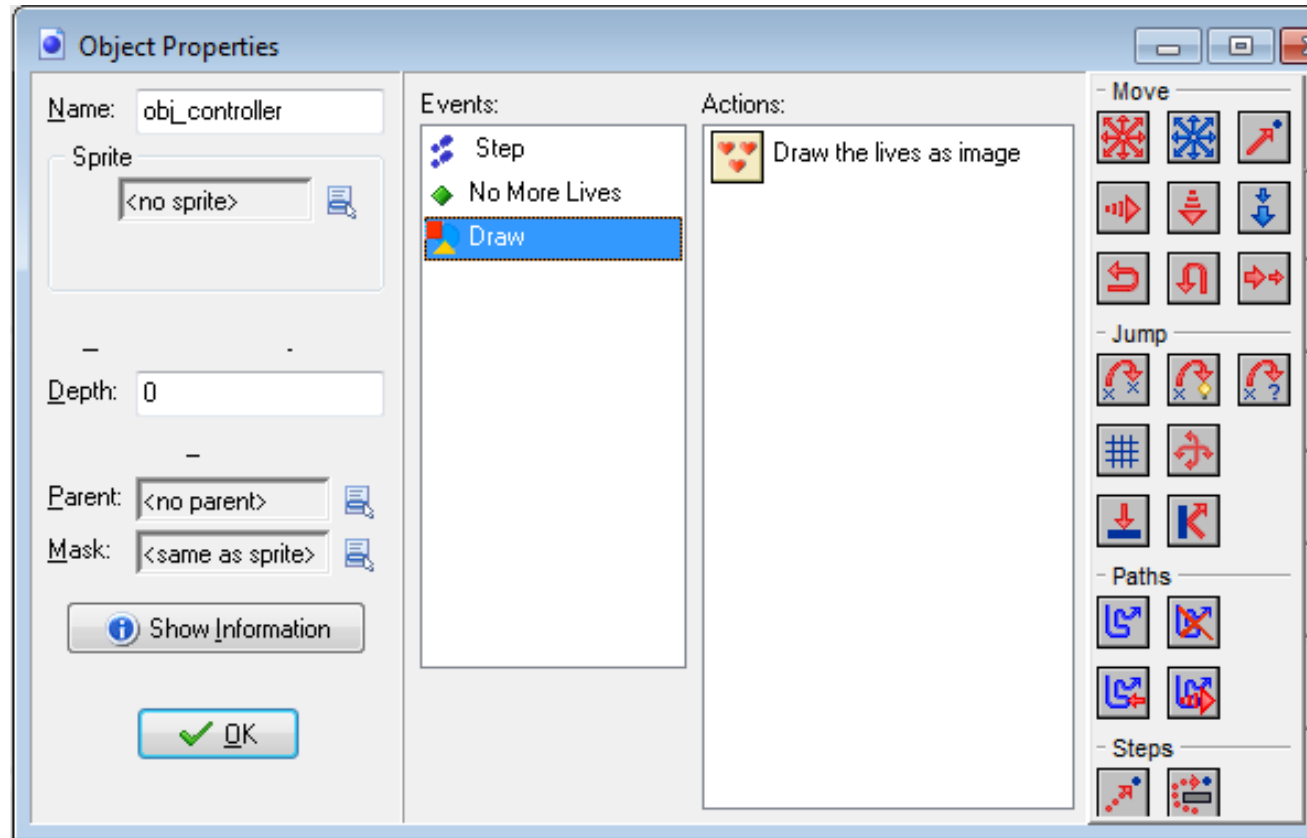


Inheriting Events

- A child inherits all the events (and actions) of its parent.
- A child can have its own events as well, but these only apply to the child and not the parent.
- When both the child and parent have the same event with different actions, then the actions of the child are used for the child, and the actions of the parent are used for the parent.

Tracking Lives

- The Draw event is used to track the number of lives Katch has left in the controller object.



Blocks

- Increasing the difficulty:
 - Solid
 - Invisible
 - Special Block – require being hit twice before destroying.

Levels and Sound

- Add the finishing touches and you are ready to play!