3

Herding Cats: Using Level Design to Tell Stories

In this chapter, you'll learn about *level design*, which is the blueprint of each level in a game. A level can be one scene in a game, one room, one area, or one part of the journey through the game. For example, each puzzle in *Herding Cats* is one level. You can imagine the whole game as a tower of levels: each one is stacked on top of the other, and the players work their way through the stack one level at a time.

Level design tells stories. It shows a player what's important and can teach the player the fundamental ideas of your game. For example, in *Herding Cats* you sometimes need to use a cat to reach another cat. You can use level design to develop those ideas; use it in trickier, harder ways; and mix it up to create something unexpected. Level design can surprise the player or make them feel different emotions, such as smart, excited, scared, frustrated, or curious.

Think of each level in a game as a kind of tiny game, each with its own challenges and solutions. The characters might be the same, but the situation changes with each new level; each new level teaches the player more about the characters and the game.

Using the Level Editor

In this section, we'll create levels for *Herding Cats* using PuzzleScript's built-in level editor. We'll identify the important ideas in *Herding Cats* and figure out how to introduce them to the player. Then we'll build on them. We'll think about what goes into a good level and what levels go into a well-paced, complete game. And we'll tell a little story while we're at it.

Let's start by building the smallest possible level using the built-in level editor. Enter the following in your PuzzleScript game's LEVELS section to build a level. Or visit *bit.ly/catswithoutlevels* for a fully programmed copy of the game with the levels missing.

```
=======
LEVELS
=======
###
#m#
#.#
#p#
###
```

This creates your first level with one cat (m), a starting position for the player (p), an empty space in between them (.), and walls (#).

Playing a Level

To play the level you just entered, you can use a handy shortcut: hold down the CTRL key on your keyboard (COMMAND key on Mac) and click the level. It should pop up immediately in the game, ready to play, as shown here.


Pretty much the smallest Herding Cats *level you can make*

Press the up arrow to wake up the kitty and win!

Editing a Level

Now let's edit the level you entered to make it more interesting. To enter editing mode, CTRL-click the level again and, while pressing CTRL, press the **E** key. A row of objects should appear that includes the letter *S,* some grass, a wall, the player, and a cat, as shown in the following figure. (The cat should just be a pair of yellow eyes, because it's a black cat on a black background.) We'll call this the level editor *palette.*
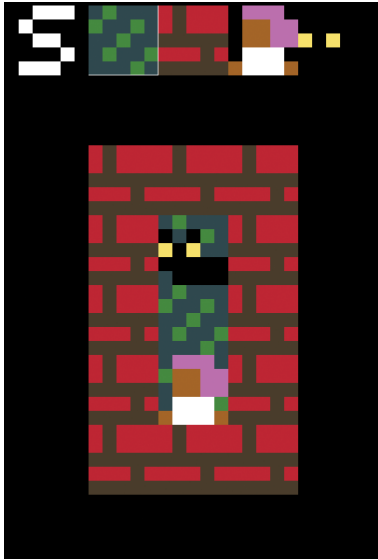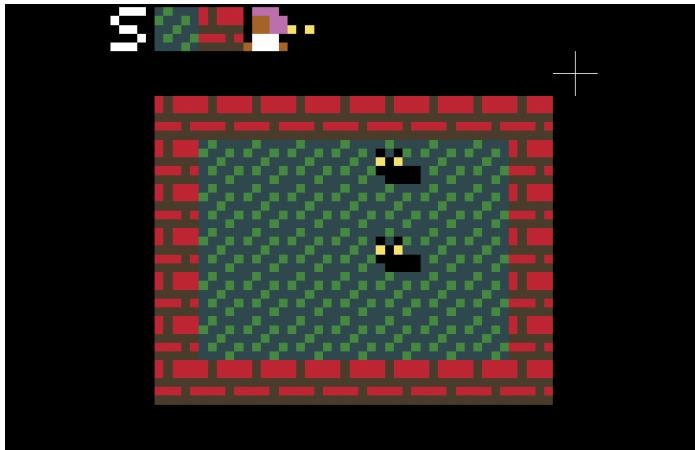
Editing mode

Click an object from the palette to select it, and then click in the level to place it. Try selecting the cat, and then click the player in your level to change the player into a cat. To erase the placed objects, right-click them.

Enlarging a Level

To enlarge the level, click an edge, and then move your cursor to the far left, far right, top, or bottom of the level. The cursor should change from a box to crosshairs (+). Click when the cursor looks like crosshairs to grow the level by one row or column in that direction. Right-click to shrink the level. Try making the level bigger and then redrawing the walls to fit the new level. Here's what an enlarged level with two cats would look like.

NOTE: To play your level while editing it, press the arrow keys. (You might have to add a new player to the game if you turned the player into a cat earlier.)

Enlarged level

Saving and Printing Your Level

Once you finish editing a level in editing mode, save it by clicking the **S** from the palette. The box below the game should now display the text version of your new level.

```
=================================
Successful Compilation, generated 12 instructions.
Printing level contents:

########
#...m..#
#......#
#...m..#
#......#
#......#
########
```

PuzzleScript levels can also be represented in text.

Note that the text version displays the code for the level as it exists at that very moment. So if you moved the player in editing mode, the code should now display the player in their new position. But if you woke up all the cats, there should be no cats shown in the code because the LEGEND section has no code for an awake cat.

Be sure to click **S** a lot when you're working on levels to make sure you don't accidentally lose your work!

Adding Your New Level to Your Game

Once you're done updating your level in editing mode, you'll need to copy and paste the updated PuzzleScript code into your game's LEVELS section.

To copy and paste the new level, highlight the text version of your level. Next, right-click and select **Copy** to copy the text of the level to your operating system's clipboard. (You won't see the clipboard; the text will just be saved invisibly.) Then right-click under LEVELS and select **Paste** to paste your new level into your game's LEVELS section! You can also drag and drop the highlighted text into your LEVELS section.

The result should look like this.

```
=======
LEVELS
=======
########
#...m..#
#......#
#...m..#
#......#
#......#
########
```

Testing Your Levels

To test your levels without totally rearranging them, press **R** (Reset). But be careful! Pressing R will reset your level to the way it was before you first pressed E to enter editing mode. If you changed your level since you started editing, pressing R will undo all of your changes! Eek!

But there's a solution to this problem: to test a level, press **E** twice (once to exit editing mode and once to reenter it), and then play your level. Now when you press **R**, the level should again reset to the way it was when you pressed E. Click the **S** button to print your level.

Be sure to play your levels a lot while you're working on them, and save them even more frequently.
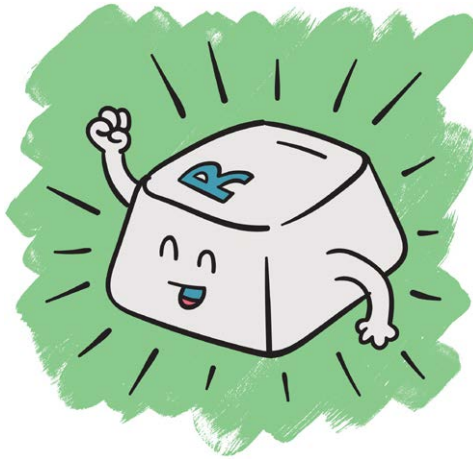
Levels Tell a Story

Let's build some levels for our *Herding Cats* game. What would be a good starting level? Your game's first level should introduce the game's story (the player tries to make friends with cats) and show the player the goal of each level, which is to wake up cats.

This figure shows a starting level that I came up with. See if you can re-create it in the editor.

Level 1: Cat friend #1
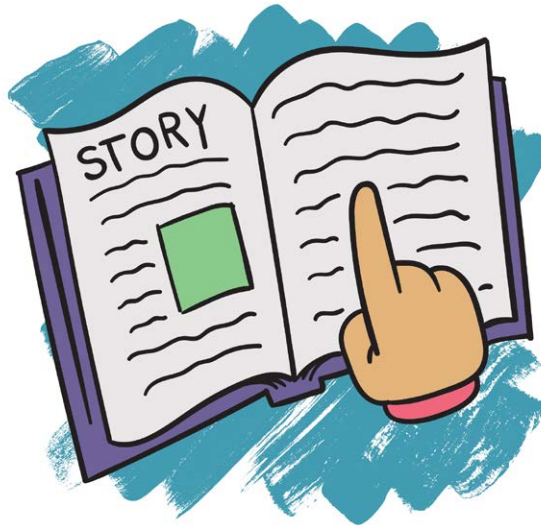
This first level is very simple: when the player touches the cat, the level ends. The player doesn't know why they're befriending cats yet. But you can help them figure this out by adding a message in PuzzleScript to give them some context about what's going on. In your game's LEVELS section, enter the word Message followed by what you want your message to be (for example, Oh, a kitty!). The words you enter in your message will display on the screen between levels (without the word Message).

```
=======
LEVELS
=======
Message Oh, a kitty!
########
#...m..#
#......#
#...m..#
#......#
#......#
########
Message Hi, kitty.
```

Let's add the message Oh, a kitty! to make it appear before the level begins at the very start of the game to establish that the kitty is important. Then let's show the message Hi, kitty. when the level ends, to tell

the player that meeting the kitty was what they were supposed to do. The player will also see some text reminding them to press X to continue the game, as shown here.


This is what a message looks like in the game.

That completes our first level for *Herding Cats*! By the end of this level, the player should know the basic point of the game.

Levels Teach the Player the Rules

We've introduced the idea of meeting cats as a game goal. The next important rule we need the player to understand is that cats will follow the player around. How can we make a level to teach the player that rule?
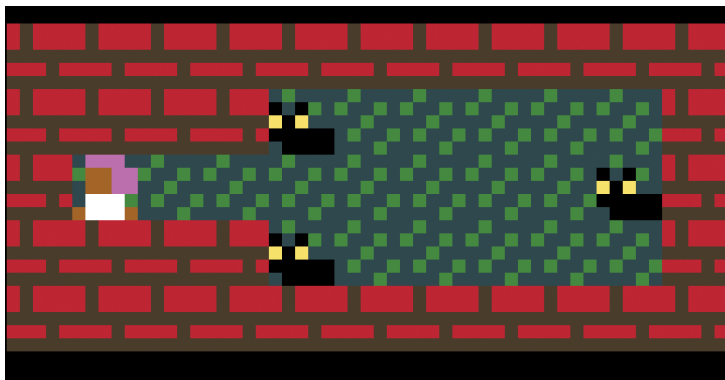

Level 2: Cats like to tag along.

To finish this second level, the player must pass by two cats to get to the third cat on the right. As they pass the cats, the cats perk up and follow. The level ends when the player has met every cat.

Troubleshooting

But what about the cat from the first level? Could that cat still be following the player? Let's modify our level to add a cat sidekick. Here's what the level looks like now.
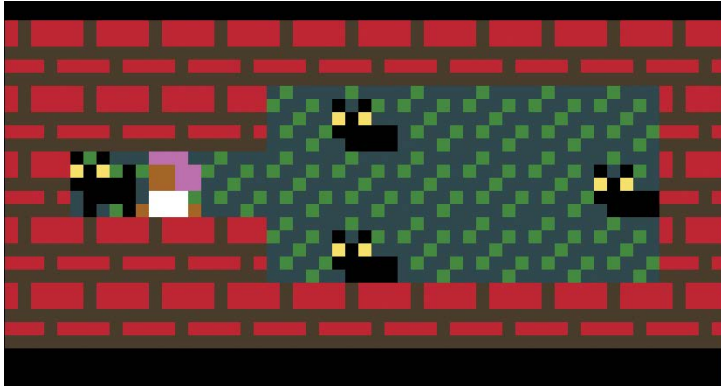
Level 2B: The player starts level 2 with their cat friend from level 1.

Build this level and try it. Uh-oh, there's a problem. Did you find it?

When you move the player, the cat from Level 1 doesn't follow. You can go back and wake up the cat, but we want the cat to be following the player *already* because this is a friend we made on Level 1!

The problem occurs because of the PuzzleScript rules we set in Chapter 2. The rule that tells PuzzleScript to turn sleeping cats into awake cats is a late rule that triggers only when the player moves. But because the player hasn't moved yet, no rules have triggered!

Running Rules at the Start of a Level

Fortunately, we can solve the problem of the first cat friend following the player by making sure all the rules run at the beginning of every level, before the player does anything. To make that happen, just add `run_rules_on_level_start` to the beginning of your PuzzleScript code, as shown here.

```
title Herding Cats
author anna anthropy
homepage www.puzzlescript.net
run_rules_on_level_start
```

Now click **Rebuild** and restart your level. The new cat should follow the player. The player should start level 2 by walking around with their cat friend from level 1. As they pass the two cats on the left, the cats wake up and start following them. Then, with an entourage of three cats accompanying the player, they meet a fourth and final cat, and the level is complete. Let's end with another message, just to make the story complete.

```
message Oh, hi to you too!
```

That's a whole little story right there!

# Levels Challenge Players to Use What They Know

So far we've taught the player that cats they touch will follow them around and that the goal is to get every cat to follow them. Now that they know those rules, let's create a puzzle that will challenge them to use what they know.

Can you make a level that forces the player to think carefully about the order in which they wake up the cats?
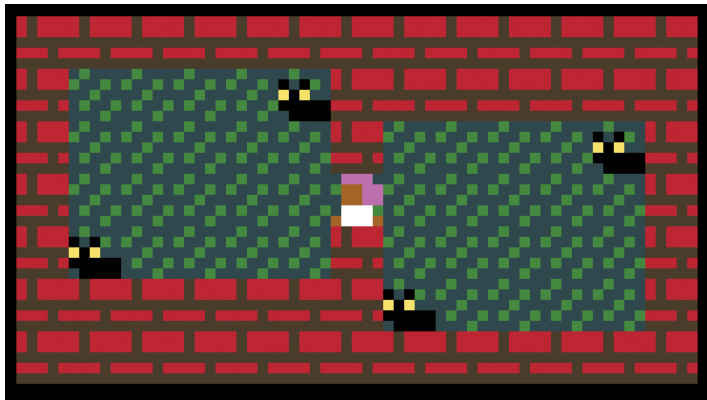

Level 3: A close fit!

Level 3 has two rooms with two cats each and a super narrow passage between them. If the player isn't careful about how they herd their cats, they won't be able to make it through the gap, as shown in the figure. Try to solve the level. Do you see how this level tests what the player has learned?
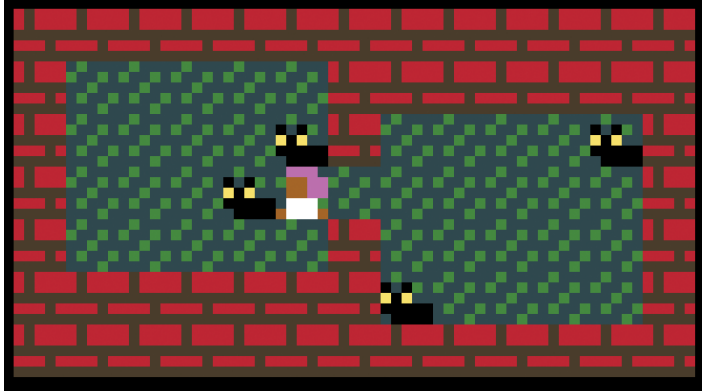
Level 3: Stuck!

If the player doesn't think carefully about the shape of their group, they'll get stuck!
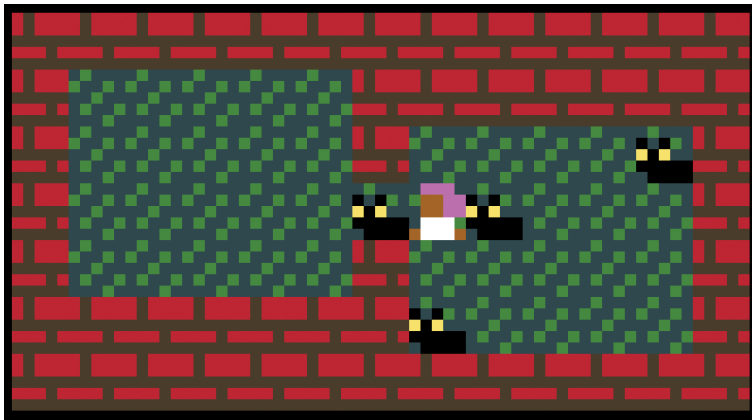
Level 3 solution

To get through this level, the player needs to arrange all the cats in the group to fit through the narrow hole.

Learning from Mistakes

Let's try to arrange the cats in Level 3 a little differently. We'll put them across from each other horizontally instead of on a diagonal. How might that affect how the player thinks about shaping their group?
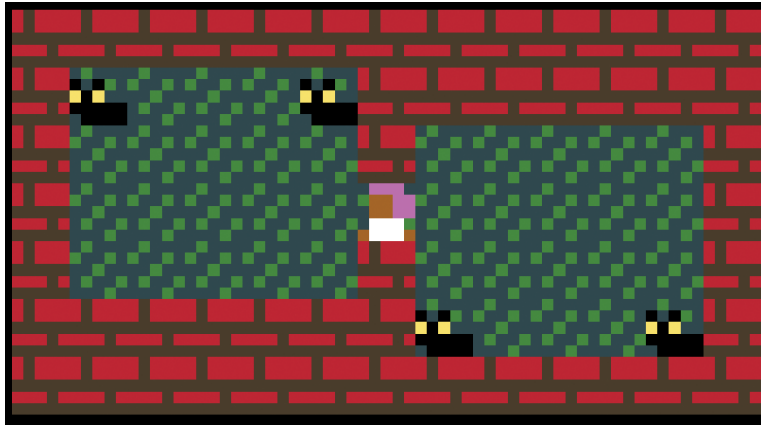
Level 3B: What if the cats were arranged differently?

Every player thinks a little differently. But it's likely that the player will have an easier time finding the solution with this setup. The cats are already lined up the way they need to be to fit through the narrow hole! So does this level change make the level better?

I would say no, because *making mistakes is an important part of play*. In *Herding Cats*, when the player makes a mistake and gets stuck, that's when they realize they need to think more carefully about where their cat friends are. When they make mistakes, they learn about how the game works and how they need to think to solve puzzles. Messing up is important!

Design Your Own Levels!

Try designing some levels of your own for the game! Here are some ideas:

- **Make a level that's harder than the previous level but only a little bit.** Then try to make a level that's a little bit harder than that! It's much trickier to make a not-too-hard level than a

very hard level. Imagine if your favorite game skipped straight to the hardest level. You would have no idea what to do: the in-between levels help you learn how to handle harder challenges! If you make a level that's too hard, think about what you could do to simplify it.

- **Make the hardest level you can in the smallest amount of screen space.** Think carefully about where you put your objects. How many do you *really* need to make an interesting level?

- **Make levels that tell stories!** When the first level ends with the player making friends with a cat and the second level starts with that cat friend still hanging out, that's storytelling! It's a simple story, but it makes the player care about the game. What other ways can you come up with to tell simple stories?

- **Make some levels with multiple players.** When you add multiple player objects, what kind of puzzles can you come up with?
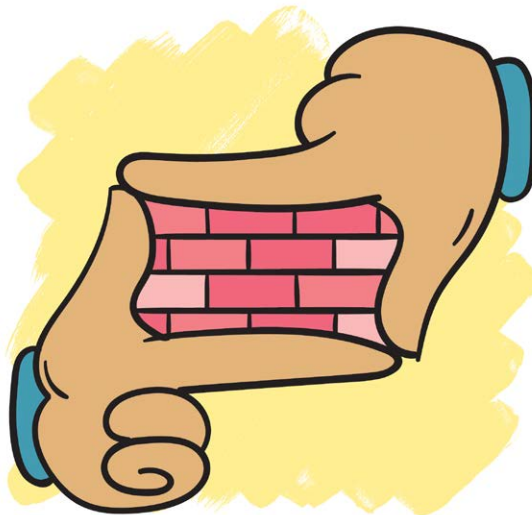
Remember to play your levels while you're working on them! Pay attention to what you're thinking and experiencing when you play each level. What is your first instinct in a level? Is it to go a particular way, and if so, why? If something about a level seems a little rough, make it better and then play it again! As you build levels, you'll develop your intuition for level design.

When you've completed your game, have friends play your levels to get a sense of how others experience a level. Remember that as the game developer, you know all there is to know about the game. But what's easy for you might be hard for someone else. It's important to see how other people react to your creations.

While you're watching someone else play your levels, pay attention to what they do. Try to resist the urge to explain to them what they're supposed to be doing or why you made a particular part a certain way. Your goal should be to get an idea of what players will do when you're not there.

It takes a lot of work to become a good level designer. Listen to your own instincts and to other people's feedback. The more levels you make, the better you'll get at it!
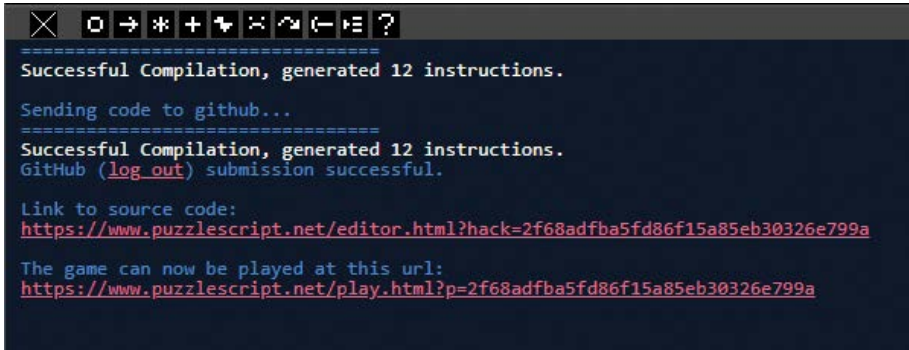
Sharing Your Game

When you've finished your PuzzleScript game, you can share it! Sharing a PuzzleScript game is super easy! Just click **Share** at the top of the screen. You'll need to create a GitHub account.

Click the Share button to create a link to your game.

When you click Share, PuzzleScript creates two links: one opens your game in the PuzzleScript editor, and the other leads to your playable game.

Share that second link with friends so they can play your game! They can also open your game in the editor by clicking the **hack** link at the bottom of the page.

*PuzzleScript generates links to your game.*

You can also save a copy of your game to your computer by clicking **Export** at the top of the screen. Exporting your game creates an HTML file on your computer that you can open to play your finished game. This version doesn't have the hack link, so if you want to keep your game code a secret, export it and upload it to a free website, such as *https://neocities.org*!

Bonus Challenges

Here are a couple of graphical tricks I added to *Herding Cats* to make the game look more interesting. Can you figure out how to do them? If you need some hints, open my finished version of the game at *https://w.itch .io/herding-cats* and check out how I did them.

Levels with Different Shapes

The first graphical trick is a simple one. Instead of being completely rectangular, the levels are all different shapes.
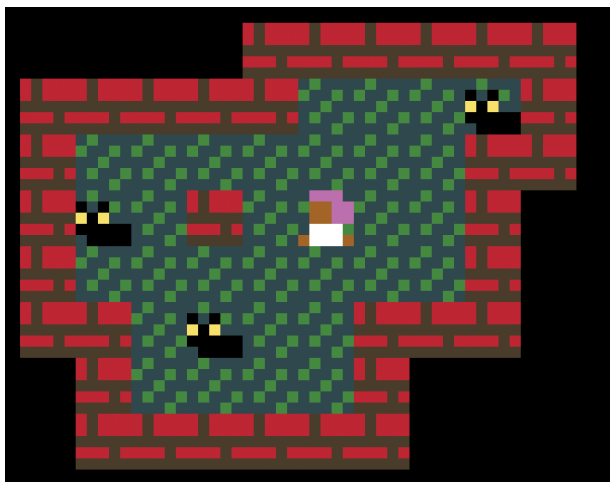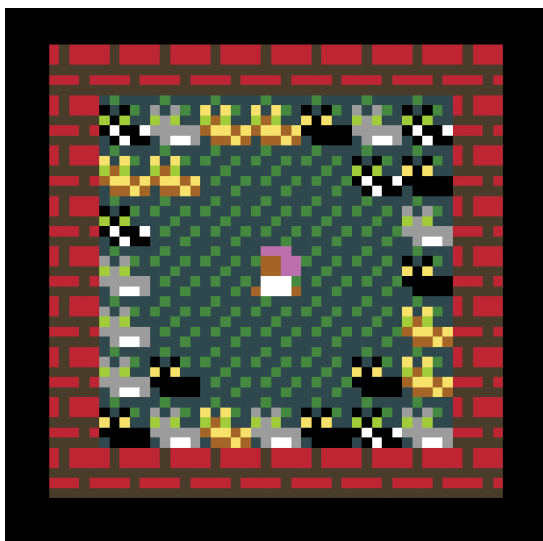
Creating levels with funky shapes

This trick is easy! I just made a totally black object and used it to fill the space outside the walls. So instead of the stages looking like big rectangles, they have unique shapes.

Four Types of Cats

The other graphical change is more complicated: instead of just one type of cat, there are four different kinds, as you can see here.

Sleeping and awake cats

This trick involves creating sleeping and waking versions of all four kinds of cats, plus a generic cat object to put in levels. At the start of a level, each generic cat will *randomly* become one of the four kinds of cats. (Read through the RULES section in my finished version of the game to see how this happens!)

One note about the editor: if you rebuild the game after adding new objects, the level editor might get confused about what order the objects are in. Whenever you add a new object, click the **Run** button to make sure PuzzleScript updates correctly. Clicking **Rebuild** is still fine when you're making changes within levels or to objects' appearances.

What You Learned

Well, that's it for *Herding Cats*! In this chapter, you learned how to use PuzzleScript's level editor to tell stories, teach the player the rules, and challenge the player's understanding of those rules.

Next, you'll make a game called *Robot Heist* and learn how to create obstacles like lasers. You'll also explore some clever ways to use the level editor to make your game more interesting. Soon you'll be a PuzzleScript master!