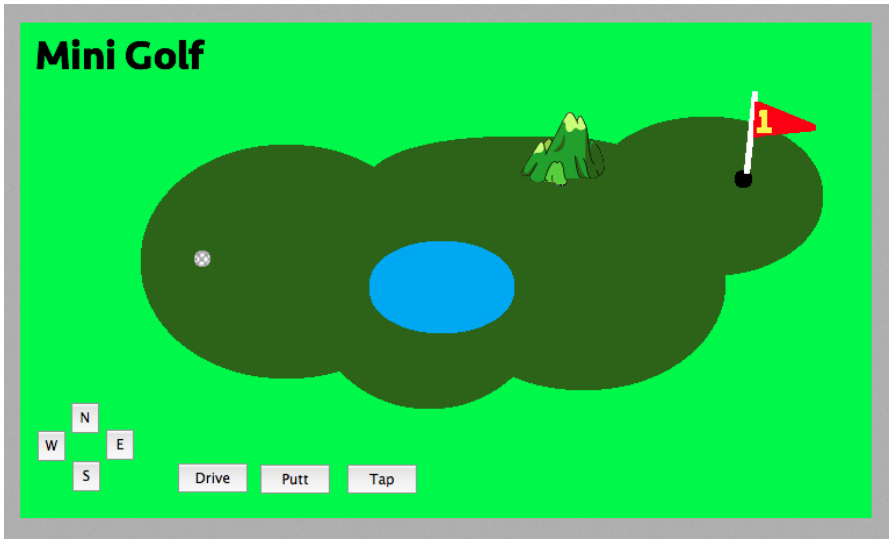
Coding For Kids Sample Project

Create a Mini Golf Game

One of the many cool things you can do when learning to code is making a game or toy. This project guides you in creating your own Mini Golf game that you can share with friends.

In the role of game-maker, you are in control of graphics tools as you paint a golf hole, and then create ball and obstacle objects. You put your button-making skills to use as you code motion and direction commands. Next, you code how the ball reacts to the environment — falling in a water trap or the hole, as well as reflecting off an obstacle.

Several simple commands are used in this golf course, plus a couple of easy-to-write procedures. When complete, consider adding new features to make your game even more challenging and fun to play!



Brainstorm

Miniature golf courses can be crazy! You can make a themed hole with all sorts of decorations and colors, as well as weird bumpers and traps. Explore the Painting/Clipart palette and go a little wild with your game theme:

- Tropical
- Spacey
- Sports

In my hometown of Las Vegas, there's a miniature golf course featuring Kiss — the 1970s rock band — where you can golf *and* rock and roll all night!

Downloading the MicroWorlds EX Software

In this project, you create the Mini Golf game using a free trial of MicroWorlds EX. The following steps explain how to get the free trial software:

1. Visit the website where you register for the trial software: www.microworlds.com/solutions/demo_ex.html
2. Read through the details about the trial, and at the bottom of the page, enter your name, country, email address, and platform (Mac or PC).
3. After you register, check your email for a link to download the free trial software.

Start a New Project

Begin creating Mini Golf by starting a new project as follows:

1. Start MicroWorlds EX.

2. On the yellow MicroWorlds EX startup screen, select Free Mode.

   A new project opens.

3. From the menu bar, choose File→New Project Size→MicroWorlds Standard.

Draw a Golf Green, Water Trap, and Hole

Paint a golf green where the player will play. Include a water trap and a target hole. Follow these steps:

1. On the toolbar, click the Hide/Show Painting/Clipart button:

   The Painting/Clipart palette opens.
2. Select the Painting Tools on the Painting/Clipart Palette:

   
3. Select a background color in the color palette by clicking it.

   
4. Click the Paint Can tool and then move into the workspace. Click in the workspace to fill the background.

   In Figure 1, a solid shade of lime green is used.

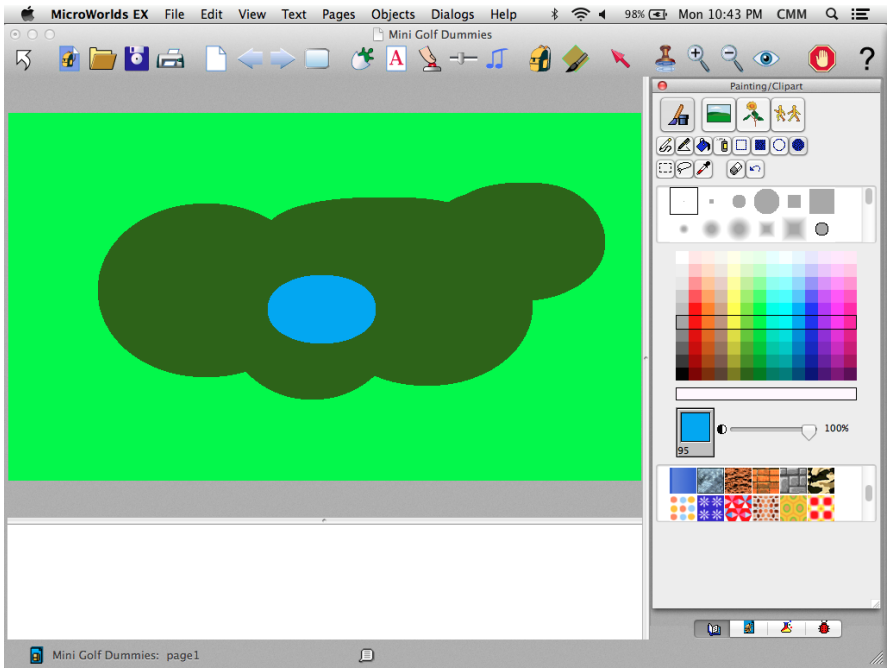


Figure 1

5. Now, create the grass for Hole 1. Select a different color in the color palette by clicking it. Click the Filled Oval tool and select a dot-sized drawing brush. Move into the workspace and draw two or three overlapping ovals, refer to Figure 1.

Tip: If you make a mistake while painting, you can undo each step by clicking the Undo button in the Painting/Clipart palette, or by clicking Ctrl-Z (Windows) or Command-Z (Mac).

6. Next, create the water trap. Click a shade of blue in the color palette. Continue painting with the Filled Oval tool and brush you selected in Step 4. Move into the workspace and paint a blue oval onto the grass (see Figure 1).

   This blue oval will be the water trap.

7. Now create the target hole. Select black in the color palette, and then select the Pencil and the small circle drawing brush. Move into the workspace and click one time near an edge of the putting green to create a circle where the golf ball will fall, as shown in the title figure (the figure at the beginning of this project).

8. Make a line for the flag post as follows: Select white in the color palette, and then select the Pen and the small square brush. Click in the hole, and then drag and release to make a line (refer to the title figure).
9. Paint a flag on the pole as follows: Select a bright color in the color palette, and then select the Pen or Pencil and the small square drawing tip. Make three lines to create a triangle for the flag on the post. Make sure the triangle is closed. Use the Paint Can to fill the inside of the triangle with color.

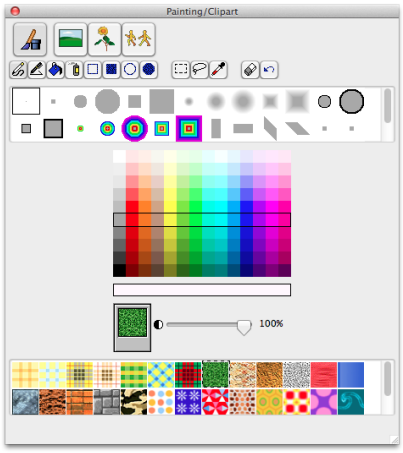
   Red is used for the flag in the title figure.

   Warning: Be sure to completely enclose a shape that you intend to fill. If an enclosure has a break in its walls, paint from the Paint Can spills out when filling the shape.
10. Number the flag as follows: Select a contrasting bright color in the color palette, and then select the Pen and the small square brush. Make lines to draw the number of the hole on the flag.

    The title figure shows a yellow number 1 drawn on the flag.

Leave the Painting/Clipart palette open. You will use it again soon.

Painting with the Painting/Clipart palette



The Painting/Clipart palette has tools for drawing and painting in the workspace and for making new shapes for turtles. It also has premade backgrounds, single shapes, and animation shapes for use in your projects. Access the Painting/Clipart palette by clicking the Hide/Show Painting/Clipart button on the toolbar. To see the painting tools, click the button in the upper-left corner of the palette.

The row of small buttons includes the drawing and painting tools (from left to right in the figure): Pencil for free-form drawing; Pen for lines; Paint Can for fills; Spray; Rectangle; Filled Rectangle; Oval; Filled Oval; Selector for selecting rectangular regions; Lasso for selecting free-form areas; Color Picker for sampling a color; Eraser; and Undo.

Below the painting tools are the drawing brush tips. Each brush is a different shape, diameter, fade, and style. You can also double-click on an empty spot to open the Brush Editor and create your own brush!

The middle of the palette consists of the paint colors. Each color has a number from 0 to 139. Each column of colors is a family. Grays are in the first family, followed by reds, and so on. Below

the color families is the Opacity slider. Set at 100%, a selected color is opaque (not see-through). Set at 0%, a color is transparent (see-through). The bottom of the palette consists of textures.

Create a Title Text Box

Add a title to the page as follows:

1. On the toolbar, click the Create a Text Box button; move into the workspace and draw a rectangle for the text box; type a title — Mini Golf — in the white area of the text box.

2. Select the text inside the text box. From the menu bar, select the Text menu options and format the text.

   See Project 1 for details on formatting text.

   Tip: You can resize text boxes at any time. Ctrl-click (Windows) or Command-click (Mac) an opaque text box. Sizing dots appear — click and drag any of them to resize the text box.

3. Right-click (Windows) or Ctrl-click (Mac) inside the text box and select Transparent from the pop-up menu.

Create a Golf Ball and an Obstacle

The golf game needs two turtle objects: one to serve as a golf ball, and another to be the obstacle. Follow these steps to create the two turtles:

1. On the toolbar, click the Create a Turtle button. Move into the workspace and click to hatch a turtle.

2. Drag the turtle to a position on the putting green. Right-click (Windows) or Ctrl-click (Mac) the turtle and select Open Backpack from the pop-up menu.

   The turtle backpack opens, as shown in Figure 2. For more details on the backpack, see the nearby sidebar, “Exploring the turtle backpack.”

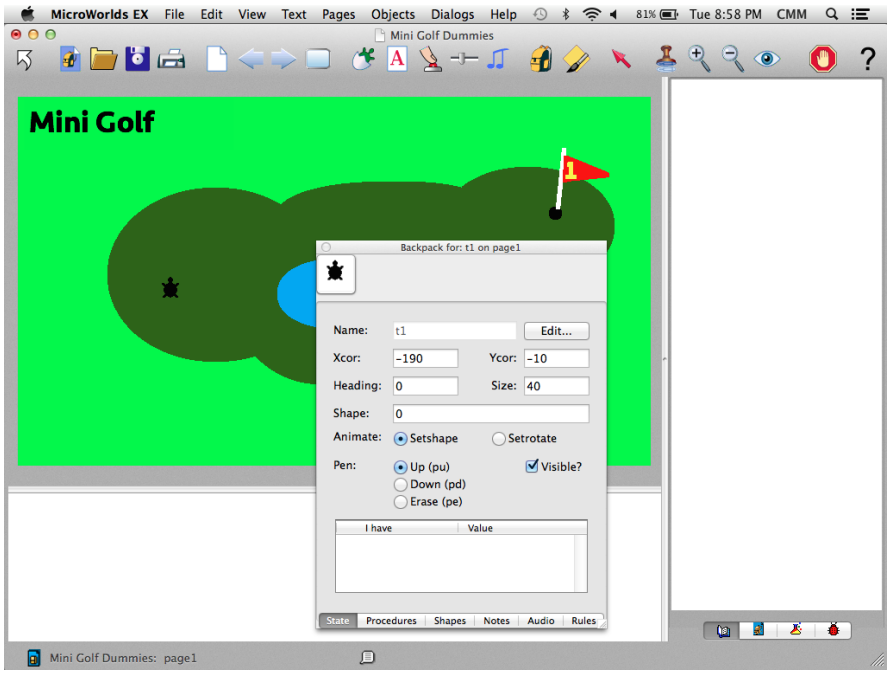


Figure 2

3. On the backpack State tab, click the Edit button beside the Name field.

   The Name dialog box appears.

4. Type `golfball` in the Name field, as shown in Figure 3. Click OK to close the Name dialog box.

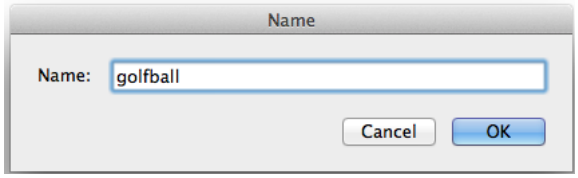


Figure 3

5. Time to find a shape for the ball turtle to wear! Return to the Painting/Clipart palette. Click the Singles button to show a collection of shapes you can use in your projects:

   Shapes in the Singles area can be worn by turtles or stamped onto the background.

6. Scroll down to the ball shape located beside the star.

   This ball shape looks similar to a golf ball, but it needs some recoloring.

7. You can recolor the golf ball on the project Shapes pane. Click the project Shapes tab (located in the lower-right corner of the window):

   The project Shapes pane holds the collection of shapes used in your project.

8. Drag the ball shape from the Painting/Clipart palette to any spot on the project Shapes pane.

   Your screen now looks similar to Figure 4.

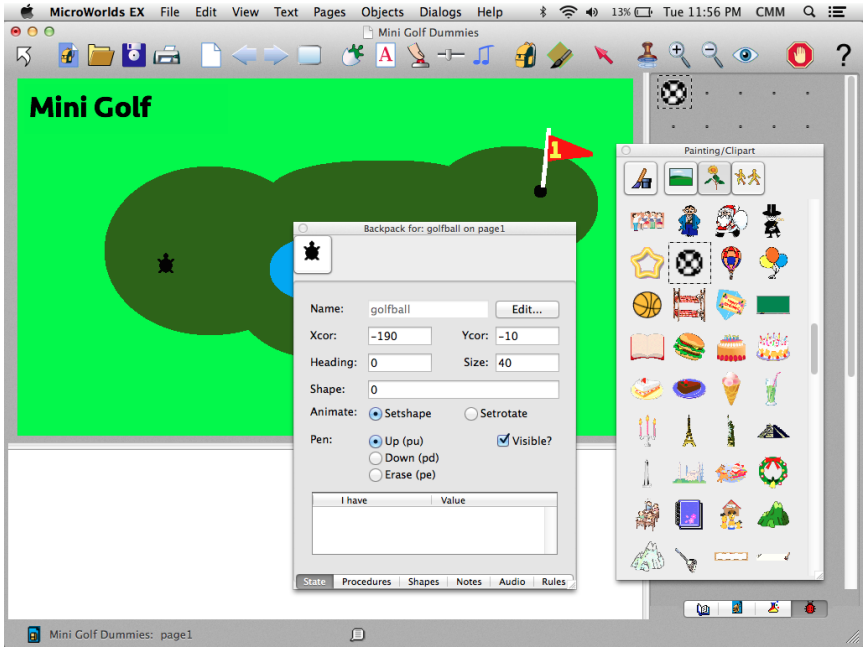


Figure 4

Exploring the turtle backpack

In MicroWorlds EX, each turtle object carries a *backpack.* The backpack holds important information, organized into six tabs (sort of like pockets):

- **State tab:** Like a wallet, the State tab includes the turtle Name, where it is located (its Xcor and Ycor), the direction it is pointed (Heading), Size, Shape, Animation mode (SetShape or SetRotate), Pen state (Up, Down, or Erase), and Visibility. Variable values known only to a specific turtle are stored in the lower I Have Value section of its State tab.
- **Procedures tab:** Holds procedures that only its turtle owner knows. This differs from the project Procedures tab, which holds procedures known by all turtles in the project. If a project procedure and backpack procedure have the same name, the turtle uses its backpack procedure.
- **Shapes tab:** Holds shapes that the turtle wears, similar to a closet of clothes. This differs from the project Shapes tab, which holds shapes wearable by all turtles in the project. If a project shape and backpack shape have the same name or same number, the turtle wears its backpack shape.
- **Notes tab:** Holds turtle-specific notes you want to keep.
- **Audio tab:** Holds music and sounds unique to the individual turtle.
- **Rules tab:** One of the most frequently used tabs in the turtle backpack. The Rules tab tells the turtle what to do when clicked on (OnClick), how to behave when walking over colors (OnColor) or bumping into other turtles (OnTouching), what to do at specific times (OnTick), what do to when hearing a message (OnMessage), and how to react to User Defined Events (When This Do That).

9. Double-click the shape spot with the ball.

   The Shape Editor opens. The Shape Editor contains the same painting tools as the main Painting/Clipart palette.

10. Select a medium shade of gray and select the Pencil and dot-sized brush. Paint the gray over the black areas to make it look more like a golf ball.

    Figure 5 shows the painting in progress.

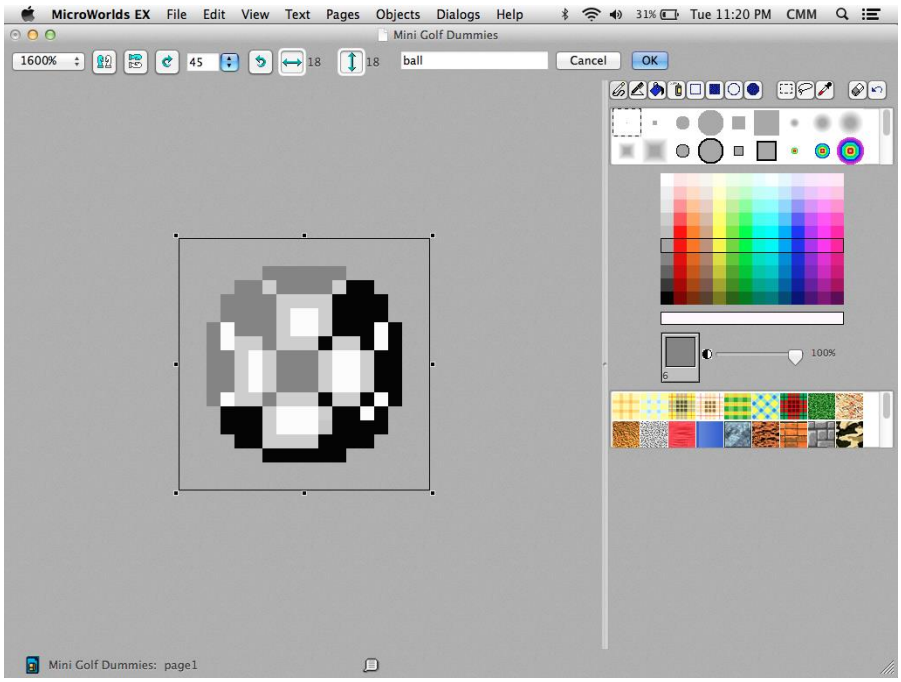


Figure 5

11. When you're finished, click OK at the top of the Shape Editor.

    The recolored ball appears at the shape spot.

12. Click the ball at the shape spot, and then move into the workspace and click the turtle.

    The ~~`golfball`~~ turtle now wears the ball shape. Your screen now looks similar to Figure 6. Leave the backpack open for later.

    Warning: Don't click a shape and attempt to drag it onto the turtle's back — you'll end up placing it on the background of the workspace. Just click-release the shape in the Shapes tab or Singles, and then click-release the turtle's back. If you accidentally place the shape on the background, right-click (Windows) or Ctrl-click (Mac) the shape and select Cut from the pop-up menu to get rid of it.

13. Repeat Steps 1–5 to create a new turtle and name it ~~`obstacle`~~. Select any shape from the Singles area of the Painting/Clipart palette to put on the turtle.

    You don't have to recolor this shape unless you want to — just click the shape and then click the turtle to apply the shape. Figure 7 shows how the screen looks now. In this example, the obstacle turtle wears the mountain shape.

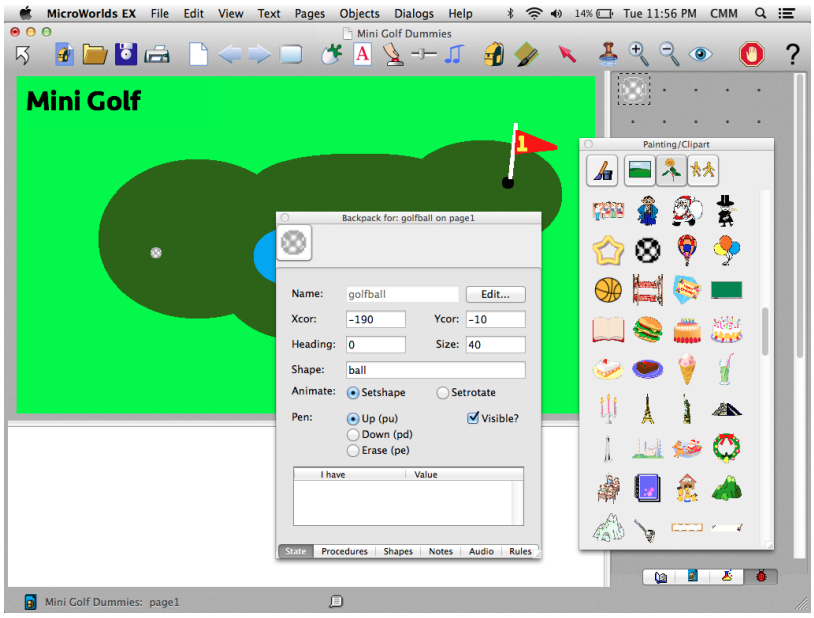


Figure 6

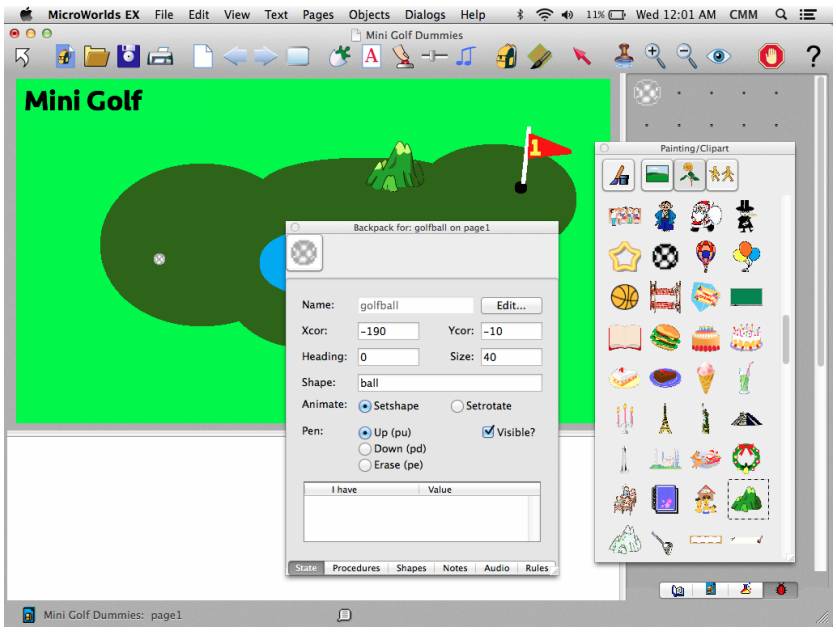


Figure 7

Set the Starting Position of the Golf Ball

At the start of the game, the golf ball must be set to the position where the player will tee off. The ball will also be set to this starting position following a fall into the water trap or a successful drop into the hole.

Follow these steps to set the starting position of the ball:

1. Drag the golf ball to a starting position.
2. Look at the State tab of the ~~`golfball`~~ turtle backpack, and note the Xcor and Ycor fields.

   Figure 6 shows that the Xcor equals `-190` and the Ycor equals `-10`. This means that the golf ball is positioned at the coordinates (`-190`, `-10`). Note that your Xcor and Ycor values may be different depending on what makes sense in your game. You will need these coordinates later in the "Write a Watertrap Procedure" section and the "Write a Win Procedure" section. Leave the backpack open.

   Math Connection: Coordinates are the mathematical way of naming a position on a graph. René Descartes is credited with the method. Coordinates in two dimensions, like your MicroWorlds EX workspace, are listed as a pair (x, y). The first number is the x-coordinate (position right to left) and the second number is the y-coordinate (position top to bottom).

Create Controls for Aiming and Hitting the Golf Ball

Aiming the golf ball means pointing it in the direction you want it to travel. Hitting the golf ball can be done in one of three ways:

- Driving it (hitting a long way)
- Putting it (hitting a short distance)
- Tapping it (hitting a tiny distance)

As the game designer and coder, you want to provide easy-to-use controls for both aiming and hitting the ball. Create buttons for pointing the ball North, East, South, or West. Then create buttons to Drive, Putt, and Tap the ball.

Coding Connection: Pointing a turtle in a direction requires turning it. Right turn (`rt`) and left turn (`lt`) are relative turn commands because they turn the turtle relative to where it currently points. Set heading, or `seth`, is an absolute turn command because it causes the turtle to point to the selected

heading regardless of where it was pointing before the turn. To use the `seth` command, you must indicate the degree angle around a circle where the turtle should point; for example, `seth 90` points east. See Figure 8.

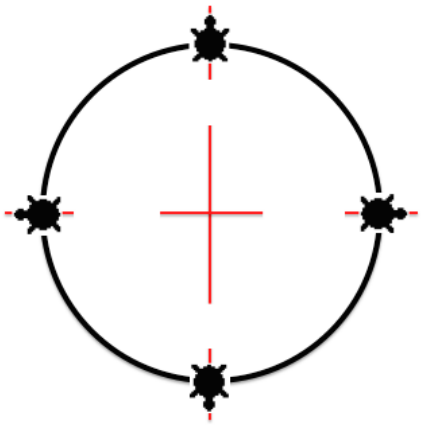


Figure 8

Create buttons for aiming the ball

Follow these steps to create the four buttons — N, S, E, and W — for aiming the ball:

1. First, create a button to point the golf ball turtle north. On the toolbar, click the Create a Button button. Then click the workspace anywhere.
2. In the Button dialog box, fill in the following information (see Figure 9):
    - *Label:* Type `N` (for north) in the Label field to name the button.
    - *Instruction:* Type `golfball, seth 0` in the Instruction field. Here's what the instruction does. The command `golfball,` (including the comma) says "I'm speaking to the golf ball." Commands that follow are executed by only the golf ball turtle. The command that follows is `seth 0`. So the entire instruction tells just the golf ball to set its heading to 0, meaning to point north.
    - *Do It:* Select the Once radio button. When the *N* button is clicked, it will execute its instruction one time.
    - *Visible:* Select this check box to leave the button visible.

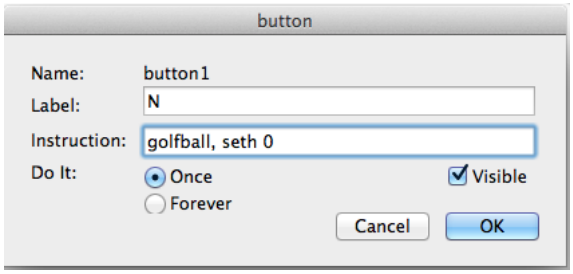


Figure 9

3. Click OK to close the Button dialog box.

   The N button is added to the workspace.

4. Drag the button to the lower-left corner of the workspace.

   Coding Connection: Arrange player/user controls such as buttons, sliders, and drop-down lists in a clear and understandable layout. Your graphical user interface, or GUI, should allow the player/user to easily operate your game, toy, or simulation!

5. Test the N button by clicking it.

   Although nothing appears to happen to the golf ball, you should see that the State tab of the golf ball backpack shows a Heading of `0`.

6. Repeat Steps 1–5 to create and test one button each for aiming the ball E (east), S (south), and W (west). Refer to Figure 8 for help in writing the instruction for each direction.

7. Arrange your N, E, S, and W buttons to match real compass headings as shown in the title figure.

Create buttons for hitting the ball

Now it's time to create buttons to get that golf ball moving! Follow these steps to create three buttons for hitting the ball:

1. On the toolbar, click the Create a Button button. Then click the workspace anywhere.

2. In the Button dialog box, fill in the following information (as shown in Figure 10):

   - *Label:* Type `Drive` in the Label field to label the button.
   - *Instruction:* Type `golfball, glide 100 0.2` in the Instruction field.

     The instruction begins with the command `golfball`, meaning that only the golf ball turtle will execute the command that follows.

The command that follows is `glide 100 0.2`, which tells the turtle to move 100 pixels at a speed of 0.2.

Tip: The `glide` primitive is similar to the `fd` primitive in that both commands cause the turtle to move. But `glide` is followed by two numbers — a distance and speed — whereas `fd` is followed by only one number — a distance. The `glide` command creates the appearance of smoother movement than `fd`.

- *Do It:* Select the Once radio button. When the Drive button is clicked, it executes its instruction one time.
- *Visible:* Select this check box to leave the button visible.

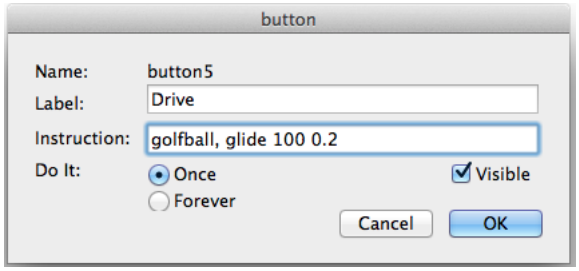


Figure 10

Coding Connection: Your button label doesn't know whether your button instruction makes sense. The button Label field could read something that in no way relates to the Instruction field! Additionally, the button Label field does not define a procedure. For example, creating a button with the label Drive and the instruction `glide 100 0.2` does not define a new procedure named `drive`. To define a procedure, write the procedure on the project Procedures pane or on the Procedures tab of a turtle backpack.

3. Click OK to close the Button dialog box.

   The Drive button is added to the workspace.

4. Drag the button to the bottom of the workspace.
5. Test your Drive button by clicking it.

   Your game's ball should mimic the motion of a real golf ball that has been hit hard! Note that the direction is not set in this command — the player must set the direction by clicking a heading button before clicking Drive.

6. Repeat Steps 1–5 to create and test one button each for Putt and Tap.

   One possible instruction for Putt is `glide 20 0.02`, but you can decide for yourself what distance and speed make sense for putt. One possible instruction for Tap is `glide 5 0.02`. Ultimately, you decide what commands work best!

7. Arrange your Drive, Putt, and Tap buttons in logical positions, as shown in the title figure.
8. Test your Putt and Tap buttons by clicking each one.

   Tip: If you want to edit the instructions you wrote, simply right-click (Windows) or Ctrl-click (Mac) a button and select Edit from the pop-up menu. The Button dialog box reappears, and you can revise your commands.

Code the Ball to Bump Off the Obstacle

Part of what makes a Mini Golf game fun is the challenge of getting past the obstacles between the tee and hole. In this game, there is one obstacle, a mountain. The player may want to hit around the obstacle, or else hit the obstacle so that the ball bounces or reflects off the obstacle.

In MicroWorlds EX, you can code an object to react to other objects that it touches. This is done in the OnTouching field of the Rules tab of the turtle backpack. Any commands or procedures in the OnTouching field of a turtle are executed when the turtle bumps into another turtle.

Make the golf ball turtle bounce off the obstacle turtle as follows:

1. In the `golfball` backpack, switch to the Rules tab.
2. At the OnTouching field, type `rt 90 glide 20 0.05`, as shown in Figure 11.

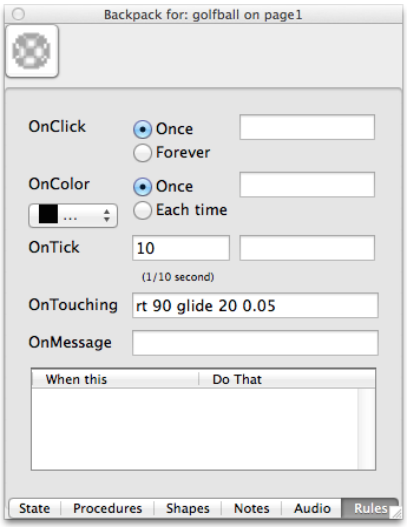


Figure 11

Here's how the commands work. The `rt 90` command turns the golf ball turtle to the right 90 degrees. Then `glide 20 0.05` moves the ball away from the obstacle by a short distance at a slow speed. The motion should look as though the golf ball is bouncing off the obstacle (the mountain).

3. Test your commands by deliberately driving, putting, or tapping the golf ball into the obstacle and looking at the result. Revise your commands as needed to create the bounce motion you want. After you are satisfied with the motion, close the turtle backpack by clicking its X button.

   Coding Connection: Collisions — two turtles bumping into each other — are common in computer program execution, especially video games! Some collisions result in a bounce, others increase a score, and others reduce the number of lives. Throughout the projects in this book, there will be lots of opportunities to code collision outcomes.

Code Universal Color Conditionals

The term *universal color conditional* is a fancy way to say, "If anything touches a certain color, this is the result." In Mini Golf, you want anything that touches the blue water trap to fall in. You will code the blue of the water trap to execute a procedure called `watertrap` when a turtle touches it. Then you write the `watertrap` procedure.

Create the universal color conditional for the water trap

Create the universal color conditional for the water trap as follows:

1. Right-click (Windows) or Ctrl-click (Mac) the blue water trap. From the pop-up menu, select Edit Sky (see Figure 12). Note that if you chose a different shade of blue than this example, your pop-up menu may show Edit Turquoise, Edit Cyan, or Edit Blue.

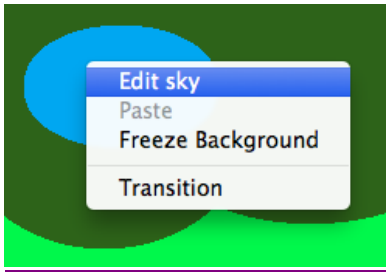


Figure 12

The Instructions for Color dialog box appears and allows you to set the universal color conditional. Because this example uses the color Sky, the name of the dialog box is Instructions for Sky.

2. In the dialog box, fill in these options, as shown in Figure 13:

    - *Mouse:* Leave the Mouse text box blank for this project. This option means that when the user clicks on the color, the associated instruction is executed.
    - *Turtle:* In the Turtle field, type `watertrap`. (You will write the `watertrap` procedure later in this project.)
    - *Once:* Select this radio button.

    This means that when a turtle is on the color sky, the `watertrap` procedure is executed once.

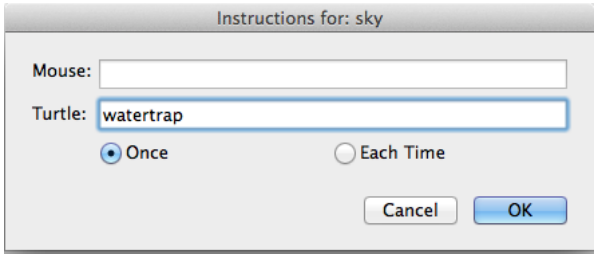


Figure 13

Warning: If you select the Each Time radio button in the dialog box, the turtle will execute the associated instruction with every step in the color until you move the turtle somewhere else. Be careful with choosing Each Time because it can cause your turtle to get stuck. There are some instances when you will select Each Time, but usually, the preferred choice is Once. Once does not mean "one time ever"; it means, "one time until you exit the color and then return again later."

3. Click OK to save your changes.

Warning: When a universal color conditional is coded, all locations on the background with that color possess the associated instruction. Be careful to look at the entire background to see where a color appears, and be especially careful of colors that appear in patterned backgrounds. If a turtle touches any instance of that color, the associated instruction is executed.

Create the universal color conditional for the hole

Create the universal color conditional for the hole as follows:

1. Right-click (Windows) or Ctrl-click (Mac) the black hole. From the pop-up menu that appears, select Edit Black (see Figure 14).

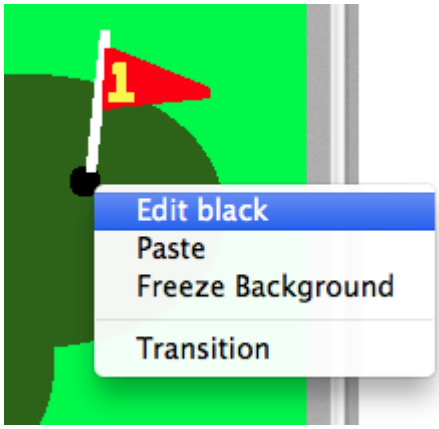


Figure 14

The Instructions for Black dialog box appears and allows you to set the universal color conditional.

2. In the dialog box, fill in these fields (as shown in Figure 15):
    - *Mouse:* Leave this text box blank.
    - *Turtle:* In the Turtle field, type `win`.
    - *Once:* Select this radio button.

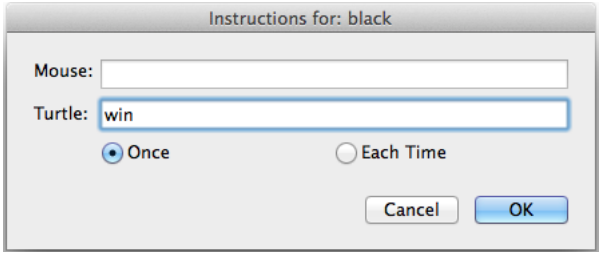


Figure 15

This means that when a turtle is on the color black, the `win` procedure is executed one time. This occurs when the player has sunk the ball in the hole! You will write the `win` procedure in the next section.

3. Click OK to save your changes.

   Tip: Universal color conditionals function only on background colors, not turtle colors and not colors in shapes worn by turtles.

   Coding Connection: Universal means that any turtle walking across the color or mouse clicking the color will cause the instructions to be executed. Conditional means that MicroWorlds EX treats the command as an `IF-THEN` command, even though the command isn't written using `IF-THEN` structure. There are also turtle-specific color conditionals: With these, each turtle is coded to individually respond to (or ignore) certain colors on the background. Universal color conditionals are coded on the background, while turtle-specific color conditionals are coded on the OnColor tab of the turtle backpack. Because there is only one moving turtle in Mini Golf, the color conditionals can just as easily be programmed in the turtle backpack.

Write a Watertrap Procedure

The `watertrap` procedure is executed when a turtle touches the water. Note that the only turtle moving in the workspace is the golf ball turtle. Follow these steps to write this procedure:

1. Click the project Procedures tab (located in the lower-right corner of the window):

2. Type the `watertrap` procedure as shown:

```
to watertrap
ht
announce [Water trap!]
wait 3
setpos [-190 -10]
st
end
```

   Here's how this procedure works:

   - This procedure starts with the `to` primitive in order to define the new command `watertrap`.
   - In the next line, it uses the `ht` (hide turtle) primitive to make it appear that the golf ball has plunged into the water.
   - The procedure then uses the `announce` primitive to issue an on-screen announcement to the player. Type the announcement you want to make between the two square brackets: `[Water trap!]`. Figure 16 shows an example of this announcement.

Figure 16

- After the player dismisses the announcement, the procedure pauses briefly using the `wait` primitive. `wait` is followed by a number that is in tenths of a second — the bigger the number, the longer the wait. The purpose of the wait is to set the pacing of the game.
- Next, the procedure sets the position of the golf ball back to the starting tee-off position using `setpos [-190 -10]` (or whatever coordinates you chose for your game). Note that because the golf ball is hidden, the player does not see it move back to the tee.
- Finally, the procedure uses the `st` (show turtle) primitive to make the ball visible again.
- As with all procedure definitions, `end` is the final primitive.

3. Drag the ball into the water and check that the `watertrap` procedure works as expected. If not, look at the Command Center for clues about possible errors.

   Coding Connection: Be sure to look carefully at the example code when deciding whether to use square brackets or parentheses in MicroWorlds EX. Other languages, such as Java, make extensive use of curly braces and parentheses. Each type of punctuation is understood differently by each computer language, very much like commands themselves. Incorrect usage of punctuation will cause program code to fail in execution.

Write a Win Procedure

The `win` procedure is executed when a turtle touches the hole. Note that the only turtle moving in the workspace is the golf ball turtle. Follow these steps to write the `win` procedure:

1. Click the project Procedures tab (located in the lower-right corner of the window):

2. Type the `win` procedure as shown:

```
to win
ht
announce [You win!]
wait 3
setpos [-190 -10]
st
end
```

You can type the `win` procedure above or below preceding procedure in the project Procedures pane. Figure 17 shows the `win` procedure following the `watertrap` procedure.

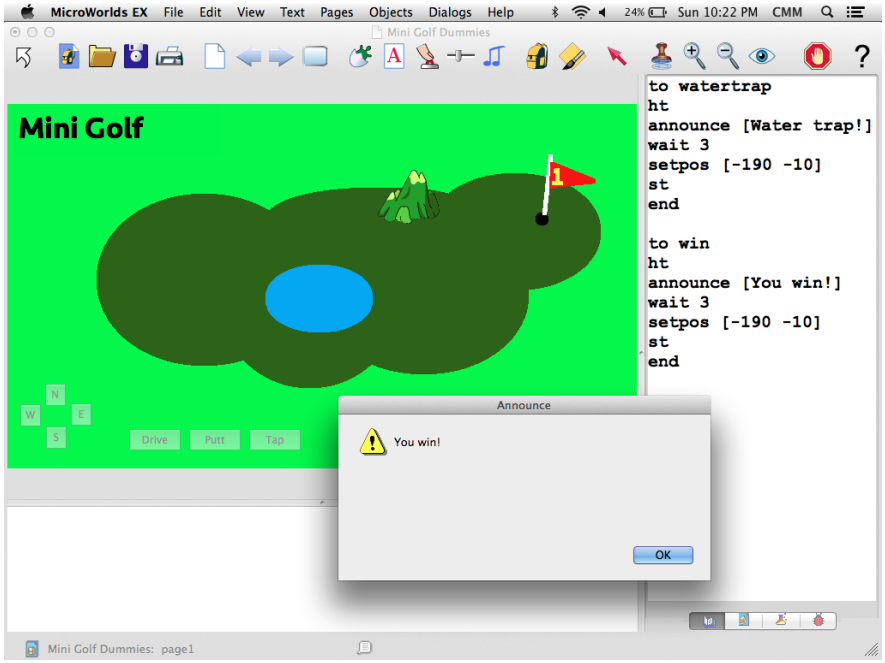


Figure 17

Note that the `win` procedure is nearly identical to the `watertrap` procedure. Only the word between the square brackets in the announcement differs: `[You win!]`. Type the announcement you want to make between the two square brackets to tell the player he has won.

3. Drag the ball into the hole and check that the `win` procedure works as expected. Figure 17 shows the completed game, as well as the successful execution of the `win` procedure.

   Coding Connection: In MicroWorlds EX, the order of the procedures entered in the Procedures pane does not matter. This is true for both the project Procedures pane and turtle backpack Procedures. However, the order in which procedures are sequenced for execution *does* matter. As an analogy, think about a pile of clothes — order doesn't matter until you need to get dressed, then it matters a great deal. Shirts go on your body before jackets, socks then shoes, and pants then belts — otherwise, your outfit receives a big error message!

Test, and Debug

Test each button to make sure it functions as you intended. Check for error messages in the Command Center to determine where any bugs may exist in your code. When you're finished, play several times and share it with friends!

Tip: Freeze the golf ball turtle, buttons, and obstacle by right-clicking (Windows) or Ctrl-clicking (Mac) it and then selecting Freeze from the pop-up menu for each element. This prevents the player from manually moving the ball and other elements of your game. Unfreeze a frozen element by selecting Unfreeze from the pop-up menu.

To view your game as a player will see it — viewing only the workspace — click the Presentation Mode button on the toolbar. The game should appear similar to the title figure. Press Esc to leave Presentation Mode.

Enhance your game

Consider enhancing your project with new features:

- **New obstacles:** Create new bumpers and other obstacles to obstruct the path of the golf ball to the hole.
- **New traps:** Consider a sand trap or a lava trap! Write a new universal color conditional procedure for each trap such as `sandtrap` or `lavatrap`.
- **New directional buttons:** Write in-between directions to aim the ball. Northeast (NE) would have an associated instruction of `seth 45`. What would the instructions be for SE, SW, and NW? Position these buttons logically with the existing direction buttons.
- **New golf holes:** In the `win` procedure, add a `nextpage` command before the `end` command. From the menu bar, choose Pages→New page to draw Hole 2! Be sure that the ball tees off from the same coordinates as Hole 1. You can make an entire 9- or 18-hole mini golf course.