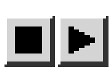
Extra Task Tutorial

In order to implement the last task, you must know about a couple more of Gamut's features. These features are not at all difficult, but they aren't really needed to complete the first two tasks, so we saved introducing them until now. The last task has two elements that are not found in the previous two. First, it contains a "monster"-like object that moves around on its own. You'll need to learn about timers to make that work. Also, the player uses the mouse to click on the screen to tell the game which way to move. This requires that you use the player input icons. This part of the tutorial will discuss these two techniques.

Timers

The timer is used to create behavior that automatically repeats at a fixed interval.

- **Open the file "path.gam". This is the file you saved earlier during the first tutorial.**
- **Create a timer widget. (The palette item looks like:**  **)**
- **Set the timer to tick every 500 milliseconds. (Hint: it's a property.)**

The timer widget has a set of three buttons below the picture of a clock. The first button is the "step" button which has this icon  . When you push this button, the timer ticks exactly one time. You use this button to demonstrate what the timer does for your game. The other two buttons which look like  are the "stop" and "play" button which turn the timer off and on.

Let's transfer the behavior of the "Combined" button you demonstrated earlier into the timer.

- **Press the "step" button on the timer.**
- **Press "Do Something."**

- **Press the “Combined” button.**
- **Press “Done.”**

Now, when you press the timer’s “step” button, it will shuffle the deck and do all the stuff you demonstrated for the “Combined” button. If you press the “play” button everything will happen automatically. Just for fun, while the timer is running, you can create more rectangles and add them to the path and they will move like the others.

Stop and Pause

In the previous tutorial, we skipped discussing the stop and pause buttons because they only affect timers. Now that you know about timers, you should also know about the rest of the history controls. Recall that the history control buttons are in the toolbar and look like the image in Figure 1.

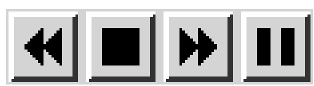


Figure 1: The time control buttons. The buttons are: back, stop, forward, and pause.

They are used to step back and forth in your game’s history as well as pause and stop timers. We mentioned the “back” and “forward” buttons ( and ) in the last tutorial. The “stop” button is labelled  . This is a global stop button that will stop all timers. You may want to turn on the timer you demonstrated above and then stop it by pushing the global stop button. The last button is “pause” which is labelled with  . The pause button is like stop in that it turns off timers except that the timers remain active. In other words, they are still “on” but they do not tick. If you turn off pause, the timers that were previously active will continue to run. Stop will just turn the timers off outright.

When you use the back button and go to a previous state, Gamut will automatically turn on pause. This is necessary because if a timer is running when you back up, it will mess up the state of your game and you would not be able to look at the history. Because Gamut sometimes activates pause on its own, it is easy to forget to turn off pause. Gamut will flash a box around the pause button whenever you push the play button on a timer while pause is active. If you do not mean for the game to be paused, you can deactivate it by pushing the pause button again.

The previous example is no longer needed and its graphics will probably be in the way for the next example. Since you already saved this example during the last tutorial we can just save into the same file. Then we'll clear some space for the new example.

- **Use the "Save" command in the "File" menu to save your modifications in the original file.**
- **Use the "New" command in the "File" menu to begin a new game.**

Player Input

Some games allow the player to use the mouse to manipulate the game. To demonstrate player input, you use the "mouse icon palette" (see Figure 2) that sits right below the main object palette. Mouse icons are created similarly to the way you would create graphical objects. First, you select the icon you want to drop and then click on the position in the window where you want to drop it. Gamut interprets a dropped mouse icon as though the player has just performed the corresponding event. If you drop a mouse click icon, Gamut reacts as though the player has clicked the mouse. There are also icons for performing mouse down, drag, and up events, as well as double down and double click, but we will only use the mouse click icon which looks like this  . You can only drop mouse icons into the blue frame window and not into the surrounding areas. That's because the player can only click on the window and cannot see any of the offscreen area.

Let's make a behavior that creates a yellow circle wherever the player clicks the mouse. Let's also make it so that when the player clicks on a circle that already exists, Gamut will delete that circle (instead of creating one).

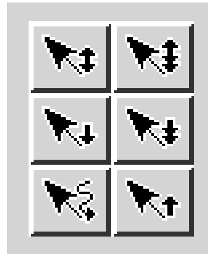


Figure 2: This palette contains all of the player input icons. The "click" icons have an arrow head pointing both up and down.

- **Drop a click icon into the window. (Its palette icon looks like  .)**
- **Press "Do Something."**
- **Create a yellow circle and place its center at the tip of the click icon.**
- **Highlight the click icon (to indicate that its position is important).**
- **Press "Done."**

This is the first time we've demonstrated creating a new object. Gamut marks created objects with a big "C" in the center. The C will remain attached to the object until you perform another event. You can test that your new behavior works by dropping more click icons in the window. You will see a C inside each new object as it is created.

Now, let's demonstrate the deleting part of this behavior.

- **Drop a click icon onto an existing circle. (Try not to put it in the center because Gamut might think that the alignment is intentional.)**
- **Select the newly created circle and press "Stop That."**
- **Delete the circle that the click icon is over.**
- **Press "Done."**

When you delete an object, Gamut draws a “D” in the place where the deleted object used to be. The D is important because this is what you would select if you need to use Stop That on a deleted object.

At this point, Gamut will want to know why it has changed from creating an object to deleting one. The reason is because the player clicked on a circle.

- **Highlight the click icon and the ghost of the deleted object.**
- **Press “Learn.”**

Here is where the time control really helps. We can use the time control to bring back the deleted object so that we can see if Gamut really knows which object it is supposed to delete.

- **Press the history controller’s back button. (Its image is .)**
- **Drop a click icon onto a circle but not the one you deleted earlier.**

As you can see from the big D and the fact that the wrong circle is gone, Gamut currently thinks it’s always supposed to delete that specific object. We can fix this using Stop That.

- **Select the “D” of the deleted circle and press “Stop That.”**
- **Delete the circle that was supposed to be deleted and press “Done.”**

Gamut will want to know how to pick the right object to delete.

- **Highlight the click icon and press “Learn.”**

Now your clicking behavior should work correctly. Test it by dropping click icons around the window. You can also generate player input directly through the regular mouse as well as with the icons. To activate “direct input mode” press the button on the frame of the blue window. Of course, you can’t select objects in the window frame while the direct input mode is active so to go back to “selection mode” press the button.

Save your work as “click.gam”.

Okay, that’s it. You are now ready for the final task.