Gamut Tutorial

Welcome to Gamut. With this tool, you can make games simply by showing the computer how the game works. To make a game, you need to first draw what the game looks like, and then you *demonstrate* how each part of the game works by directly modifying the game's graphics. In this study, we will provide most of the graphics for creating a new game. It's your job to make the game work.

This tutorial will show you how to use part of Gamut. First, we will show how to draw graphics and widgets on the screen. Then, we show the basics for demonstrating a new game behavior. Finally, we will show some techniques that will make demonstrating some behaviors much easier.

How To Use This Tutorial

First, read the introductory text to each section. This will tell you the basic concepts that you will be taught in that part. At the end of each section, there is a set of bulleted instructions that you must do. The results of these instructions feed into the instructions of following sections; so, please make sure you finish the instructions of one section before moving to the following section.

- **This is an instruction. Relax, have fun, and make sure you ask if you have any questions.**

Drawing Stuff

To draw an object, select the object from the palette. Then click and drag the mouse to create the object where you want it. After you draw the object, the palette will revert back to selection mode.

Note: The big blue frame in the middle of the drawing area is the window for your game. Everything you draw inside the blue frame will be visible in your game and everything drawn outside is hidden when the game is played.

In order to change an object's color or edit an object in other ways, you must select the object. To select an object click on it with the mouse. To select multiple objects, hold down the *shift* key and click on the other objects in the set. You can also “lasso” multiple objects by clicking down the mouse in a vacant part of the window and dragging the dotted rectangle around all the selected objects.

- **Draw a rectangle and a circle. (Make their size about an inch across.)**
- **Select them both and duplicate them. (The “Duplicate” command is in the “Edit” menu.)**
- **Delete one of the rectangles and both of the two circles using “Cut”.**

To change an object’s color use the “color selector” that is in the toolbar region of the window. It is located near the top of the window and looks like the image in Figure 1. There are two things that look like the color selector in

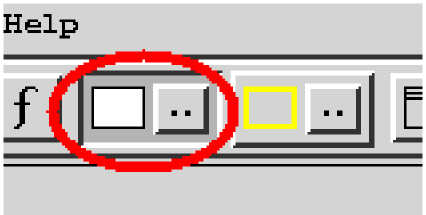


Figure 1: The color selector. This shows the color Gamut uses to draw new objects. Push the “..” button to change the color.

the toolbar. The one you want is on the left. The color selector has two parts. The rectangle in the middle shows what color Gamut will use when new objects are created. There is also a button that brings up a dialog box which you can use to change the currently selected color.

- **Change the rectangle’s fill color to red.**

When you are building a game, there are often a lot of features in the game that affect the game’s behavior but should not be visible to the player when the game is running. You can build these “invisible” objects by making them “guide objects.” Guide objects are just like regular objects except they use a color from the “guide object color selector” instead of the regular selector. The guide object color selector is on the right of the regular color selector and works the same way. All guide objects can be made invisible using the "Hide Guide Objects" command in the "View" menu.

Note: When Gamut creates a new object it can only use one of the two kinds of color. You can tell whether Gamut will use the regular color or the guide object color because the region around the indicator's box will look pressed. You can change which color is used by clicking on the indicator.

- **Make a path out of five arrow lines. Connect the start of one line to the tail of the next as in Figure 2. (Draw the arrows inside the blue window frame because you will need to have it this way later.)**

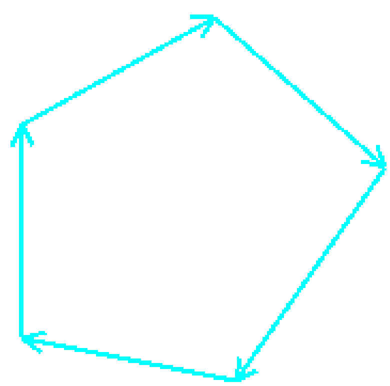


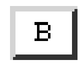
Figure 2: Draw a path of arrow lines like this.

- **Set the color of the arrow lines to be cyan "guide objects."**
- **Move the rectangle so that its center is positioned at the end of an arrow line. You may need to resize the rectangle so that you can place its center exactly at the end of an arrow.**
- **Use the "Hide Guide Objects" command in the "View" menu to make the guides invisible.**
- **Use "Show Guide Objects" to make them visible again.**

Finally, Gamut has a number of "widget" objects like buttons and check-boxes. Some widgets, like buttons, have a default size so no matter how big you draw the region, they will always be the same size when you create one.

You cannot set a widget's color so you cannot turn one into a "guide object." However, many widgets have properties that you can change. To change a widget's properties you can double-click the mouse over the object. You can also select the object and use the "Edit Properties" command in the "Edit" menu.

Selecting a widget can be a little tricky. For instance, a button can be pushed as well as be selected. Clicking in the center of a button pushes it, clicking on the edge of a button selects it. You can also "lasso" the button in order to select it. You will have to unselect the button in order to push it again.

- **Create a button. (Its palette icon looks like .) Make the label of the button read "Move."**

Demonstrating Stuff

You can teach Gamut new behavior by "demonstrating" your game's responses in the editor. In order to teach a new behavior, though, you have to get Gamut's attention. Gamut normally does not watch what you are doing while you draw on the screen. To get Gamut's attention, you use the "Do Something" or "Stop That" buttons at the bottom right corner of the window which look like the image in Figure 3. As your first example, you will

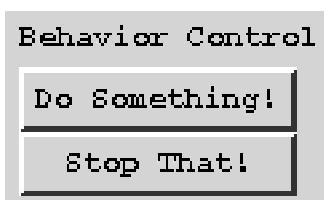


Figure 3: The Do Something and Stop That buttons. Use these to demonstrate new behavior.

make the rectangle that you created previously follow the arrow lines when you push the "Move" button.

The first thing you need to do is perform the action that causes the intended behavior to activate. In this case, you would push the "Move" button because that is what tells the rectangle to move. Then, when you see that

Gamut does not perform the right behavior, you press the “Do Something” or “Stop That” button to tell Gamut what it should have done.

- **Push your “Move” button. (That’s the button you created above.) If the “Move” button happens to be selected, you will have to unselect it in order to push it.**
- **Push “Do Something.”**
- **Move the rectangle to the next link in the path. Make sure it is centered correctly because Gamut can get confused if the picture is messy.**

We haven’t quite finished telling Gamut what to do. First, it is a good idea to show Gamut what objects are important to the intended behavior. In this case, we want Gamut to notice the path that the rectangle is following. Gamut can be a little dense sometimes and not see very obvious relationships (like when two objects are connected). You help Gamut see these relationships by *highlighting* important objects. You highlight an object by clicking on it with the rightmost mouse button.

- **Highlight the line connecting the rectangle to its original position. The screen should now look something like Figure 4.**
- **Push the “Done” button.**

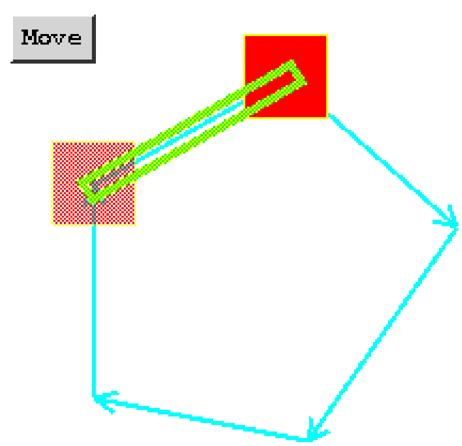


Figure 4: This is what the area near the rectangle should look like right before you press the “Done” button and finished demonstrating.

Now, when you push the “Move” button, the rectangle should follow the path just like you intended. Note: Gamut is not smart enough to learn to move the rectangle smoothly, yet. Do not be worried that objects tend to jump from one location to the next.

You can make more behaviors by continuing to show Gamut more examples. Let’s make another behavior that changes the color of the rectangle back and forth between red and blue.

- **Make a new button and label it “Color.”**
- **Push the “Color” button.**
- **Push “Do Something.”**
- **Change the color of the rectangle to blue (keeping the line style the same).**
- **Push “Done.”**

At this point, we have a behavior that changes the color of the rectangle to blue, but that’s all. Now when you push the “Color” button you’d want the color to change back to red.

- **Push the “Color” button. (Nothing seems to happen.)**
- **Push “Do Something.”**
- **Change the rectangle’s color back to red.**
- **Push “Done.”**

Now, Gamut has seen you do the same thing twice, you pushed the “Color” button each time but the behavior had different results. Now, Gamut wants

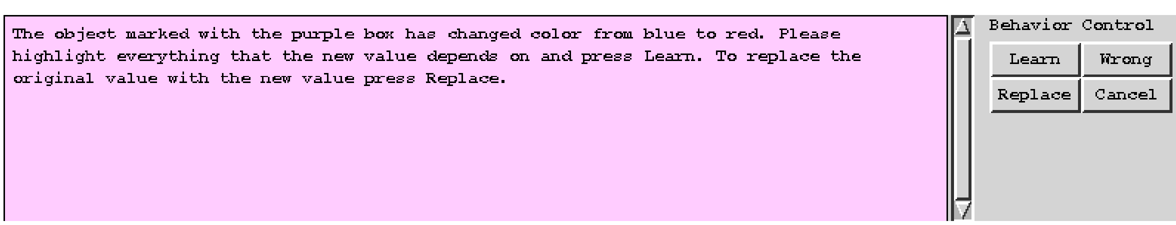


Figure 5: The dialog that Gamut uses to ask you questions. The buttons you use to answer the questions are at the right.

to know what the different colors depend on and it asks you with the dialog box at the bottom of the screen which looks like Figure 5. Alongside the text are four buttons labelled “Learn,” “Replace,” “Wrong,” and “Cancel.” **Understanding these buttons is extremely important.** The buttons are used to respond to Gamut’s questions so it is important that you know what you are telling the system. Here is what these buttons are used for:

Learn: This is the typical button you use. First you highlight the object(s) in the game that answer Gamut’s question and then you press this button to make Gamut “Learn” how to use whatever you highlighted.

Replace: This button is used to fix mistakes. You may have accidentally changed the game in the wrong way while Gamut was watching. When you push this button, Gamut will “Replace” whatever behavior it used to do with the new behavior.

Wrong: This button is used to skip dialogs that do not apply to the behavior you want to create. Make sure you know what the dialog text is asking before you dismiss it, though.

Cancel: Sometimes you will realize that you are not ready to demonstrate the behavior you just started. You can press “Cancel” to take you back to the normal editing mode without creating a new behavior.

In this case, we have not made a mistake and Gamut has not asked an unreasonable question, so we must highlight an object in the scene that tells Gamut when to make the color red or blue. The object you want to highlight is the faded “ghost object” of the rectangle which shows what the rectangle looked like before you changed its color. The ghost object looks like the image in Figure 6.

- **Highlight the “ghost object” of the rectangle.**
- **Press “Learn”**

Now that Gamut knows that the original properties of the rectangle are important, it creates the correct behavior. When you push the “Color” button, Gamut will switch the color back and forth as you intended. Of course, sometimes Gamut will not know which properties of the highlighted objects you want the behavior to use. In these cases, you will have to give Gamut more examples to sort out which properties are more important than others.

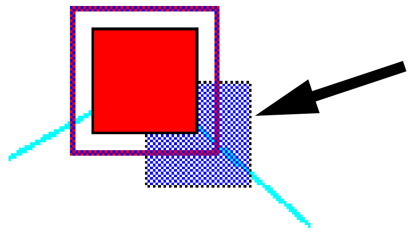


Figure 6: This shows the rectangle along with its “ghost object” which is the blue faded rectangle that is offset to the lower right from the original.

Knowing which object to highlight can sometimes be tricky. Let’s augment the “Move” button’s behavior so that it will move multiple rectangles.

- **Create a duplicate of the rectangle and place its center on the end of one of the arrows.**
- **Press the “Move” button. Here the original rectangle should have moved but not the new one.**
- **Use “Do Something” to demonstrate that the other rectangle should move as well. Press “Done” when you finish.**

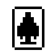
At this point, Gamut will want to know what the two rectangles depend on. Essentially, you want to move all the rectangles in the window, so it is the window object that describes both of the rectangles. You might not have thought that the window could be highlighted. You can highlight the blue window frame by clicking with the right mouse button near the inside edge of the frame.

- **Highlight the window frame.**
- **Press “Learn”**

At this point, you might be wondering what the “Stop That” button does. We’ll use it in the next example, but first we will create some new objects that we can use to demonstrate more behaviors.

Cards and Decks

Just like you can draw graphics and widgets in the main window, you can also draw stuff into “cards.” Cards are like extra drawing areas where you can put stuff that relates to an object in your game. You can get at the objects inside a card by double-clicking on it or using the “Edit Properties” command. This will bring up a special editor that works just for that card.

- **Create a card. (Its palette icon looks like  .)**
- **Open up the contents of the card using the card editor.**
- **Draw a text object inside the card (the icon looks like T) and make it read “Color” using a large, bold font.**

You’ll notice that there is a raised region in the card editor. This is the card’s “visible area.” The visible area is like a window: everything you draw in the visible area will be seen in the card object itself. You can select and move the visible region around if you want.

Stuff that you draw inside a card is difficult to get at when you are demonstrating behavior, so Gamut gives you a shortcut that lets you pre-highlight objects inside a card.

- **Pre-highlight the “Color” label by selecting it and using the “Highlight Item” command in the card editor’s “Edit” menu.**

Your card should look like the image in Figure 7. Now, whenever you highlight the card in the main window, the label in the card will also be highlighted.

- **Close the card editor for your new card. (Use the “Close” command in the card editor’s “Card” menu.)**
- **Select the card in the main window and make a duplicate of it.**
- **Open the new card’s editor and change its label to read “Move.”**
- **Close the card editor for the “Move” card.**

You now have two cards: one labelled “Color” and the other “Move.” Now, let’s make a deck out of these cards. The deck object holds a list of objects.

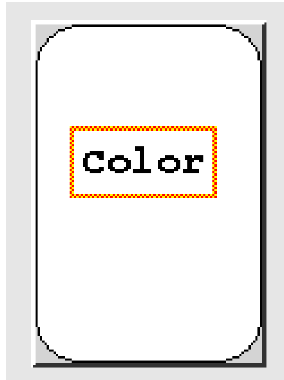
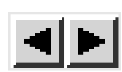


Figure 7: The “Color” label is placed in the visible area to make it visible on the card and it has a mark around it to show it is “pre-highlighted.”

To put an object in a deck, select the object and drag it over the top of the deck. When the deck accepts the object, it will show a purple outline around itself. To take an object out of the deck, select it and drag it away.

- **Create a deck. (Its palette icon looks like  .)**
- **Drag your two cards into the deck.**

You will notice there are four buttons along the bottom of the deck object. The first button,  , is the deck’s “lid.” When the lid is closed, the deck is locked and you cannot put objects in or take objects out of the deck. The second button labelled “shuffle” will shuffle the contents of the deck to randomize their order. The other two buttons,  , are used to look at objects in the deck without changing their order.

- **Close the lid on the deck and move it to the side of the window frame’s region. The main window should now look something like the image in Figure 8.**

Okay, now we are ready to demonstrate more behaviors. In this next behavior, we will use the behaviors we demonstrated for the two buttons and let the deck choose which one to apply at random. When the deck shows the “Move” card, the rectangles should move. When the deck shows “Color” the one rectangle’s color will switch.

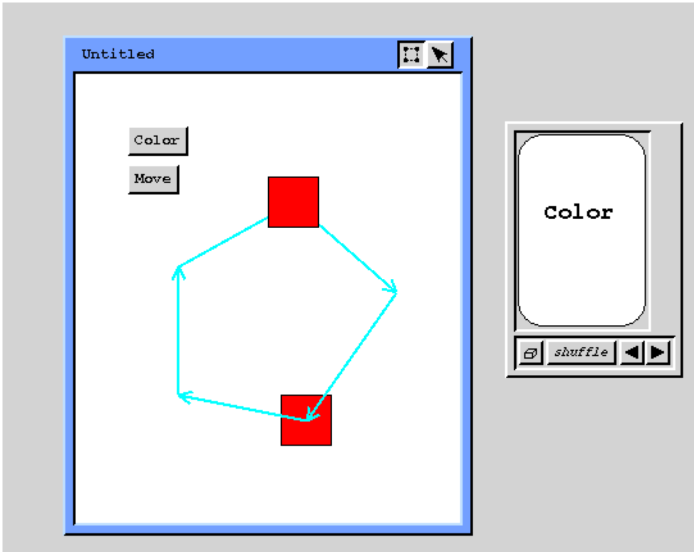


Figure 8: The main window with the assembled deck.

- **Create a new button, label it “Combined.”**
- **Push “Combined.”**
- **Push “Do Something.”**
- **Push the “shuffle” button on the deck.**

This shows that you want to deck to shuffle each time you push the “Combined” button. At this point, what you do depends on what card comes up on top.

- **If the “Move” card is on top, push the “Move” button. Otherwise push the “Color” button.**
- **Push “Done.”**

Now that your new behavior is partially defined, pressing the “Combined” button will keep moving or recoloring the rectangles depending on the random shuffle. Since you never demonstrated the other action, though, Gamut will not know what to do when the other card shows up.

- **Keep pushing the “Combined” button until the opposite card shows up.**

Finally, here is a case where you need to use “Stop That” instead of “Do Something.” You use the “Stop That” button when Gamut does something wrong like choosing the wrong behavior for the rectangles. Use “Stop That” to tell Gamut to stop performing whatever behavior it did to one or more objects. In this case, you select the rectangles and push “Stop That.” Gamut will undo whatever behavior it performed on those objects. Sometimes people will forget to select the object they want to have stopped. A handy way to remember is to say “You, Stop That” and when you say “You,” you select the misbehaving object.

- **Select the rectangle(s) that you want stopped and push the “Stop That” button.**
- **Push the button for the behavior you really wanted to have happen and push “Done.”**

Gamut will want to know why the behavior has changed. In this case, the value of the top card on the deck is what is important.

- **Highlight the deck.**

Because you already pre-highlighted the label inside each card in the deck, Gamut will automatically know that it is the label that is important. Otherwise, you would have to open up the card editor and highlight the label manually. (Gamut is not smart enough to dig around inside your cards to see what is interesting.)

- **Press “Learn.”**

It turns out, Gamut will pick a reasonable rule out of three objects you highlighted when you highlighted the deck (the top card and the text object are also highlighted). When you push the “Combined” button, the behavior should work as you intended. Gamut is not usually so easily persuaded to perform the right behavior. You should test anything you demonstrate until Gamut either makes a mistake or you are satisfied that it works. Extra examples will help Gamut sort out the properties of the highlighted objects. While Gamut sorts out different possibilities, it will accept new examples without asking you to highlight things.

Controlling History

There is one more interaction you should learn how to use. In the toolbar, the four buttons furthest to the right are the "time control." They look like the image in Figure 9, and they are used to step back and forth in your game's

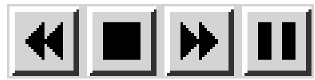


Figure 9: The time control buttons. The buttons are: back, stop, forward, and pause.

history and are also used to control timers. You will not need to use a timer in your current tasks so we will skip the "Stop" and "Pause" buttons (labelled ■ and ❚❚) for now. Just so you know, they are used as global stop and pause buttons that would affect all timers in your application.

Starting from the left, the first button is "back" which is labelled with ◀◀ . This button sends your game backward one step in time. The other button that you will want to use is labelled ▶▶ , and is the button for moving forward in time. You might think of the back and forward buttons as acting similar to undo and redo except that if you have demonstrated a new behavior during one of the steps, the behavior will still be there after you have stepped back in time over it. Using the back button is a good way to go back to a previous state in order to test what you just demonstrated or to demonstrate a slightly different example.

- **Practice using the "forward" and "back" buttons to see how it affects your game's objects.**

That completes the tutorial example. Before you go, make sure that you save your work.

- **Use the "Save" command in the "File" menu to save your work. Name the file "path.gam".**

When You Have Trouble

Unfortunately, Gamut is just a prototype system. You can make a lot of good games using this system, but sometimes Gamut can fail. For some reason, you might run into a bug in Gamut which won't let you finish demonstrating a behavior. You can use these techniques to get yourself out of a mess.

Undo: First, you should realize that undo exists in Gamut. The "Undo" command is in the "Edit" menu. For instance, if you press the "Done" button but you forgot to highlight something, you can perform undo, highlight what you want, and hit "Done" again. Note that this is not the same as using the back button in the time control which doesn't remove behaviors.

Delete Behavior: At some point, you may completely lose track of what you were demonstrating. Sometimes it can be easier to start over than to try to correct a behavior to make it work. To delete a behavior, select the object that causes the behavior to occur. This object will likely be the button or timer that you would activate to make the behavior happen. Then select the "Delete Behavior" command in the "Behavior" menu. Gamut will erase the behavior from that object and you can start over again.

Starting Over: If Gamut crashes or you are really having trouble building one of the trial games, you can always start over. The first part of each experiment will ask you to load a file as your starting point. You can restart by reloading that file. You can also save your progress if you get the game partially working and start over from that point. Please do not feel as if you have to keep going if you get lost or Gamut starts acting very poorly. Sometimes a fresh start will clear away the cobwebs and you can make the game much more easily the second time. We do ask that if you start a trial over again that you save the work that you are abandoning. This information will help us fix the problems you encountered and might help make Gamut a better tool.

Let's Begin!

Thanks for helping us with this study. When you feel comfortable with the material in this tutorial, we will be ready to begin.

Review Questions

1. **How do you demonstrate a new behavior?**

2. **How do you fix a behavior that's not doing the right thing?**

3. **How do you add new actions to a behavior that already exists?**

4. **What is the difference between highlighting an object and selecting an object?**

5. **What is the difference between using the Learn and Replace buttons when Gamut asks you a question?**

6. **What should you select before you push the Stop That button?**

7. **When Gamut asks a question, what is the most likely way it should be answered?**