# Tutorial 3 – Events

*Before starting this tutorial, you must complete the last one and have it working. Failure to do so will result in not being able to complete these steps.*

This tutorial implements event driven programming to our SDL program. So far we have had to rely on the SDL_Delay() function to enable us to see our window. Obviously when making a game you wouldn't want the screen to close after 5 seconds, and this is where events come in.

SDL supplies a good range of events for us to deal with. The ones we are interested in are:

SDL_QUIT,

SDL_KEYDOWN,

SDL_KEYUP,

SDL_MOUSEMOTION,

SDL_MOUSEBUTTONDOWN,

SDL_MOUSEBUTTONUP.

The following tutorial will demonstrate how to use events to close the window when the user clicks the X at the top right of the window.

1. As always, it recommended that you create a new branch for each tutorial. Push and merge your last branch (provided everything is working) and sync if using the VS GitHub extension. Checkout on the main branch to ensure it matches your branch from tutorial 2. If everything is in order, create a new branch called EventsTut3 (or whatever naming convention you have chosen, again, branches and using GitHub is optional but the use of and practice with will benefit you massively in future).
2. We will be creating a new function to begin with called Update, this will be our game loop that deals with events. Add a function prototype for Update() that takes no parameters and returns a bool then add the function body below CloseSDL().
3. Before we add code to our new Update function, we need to make some minor changes to the code in our main.

a. Replace the SDL_Delay code with the following:

```
//flag to check if we wish to quit
bool quit = false;

//Game Loop
while(!quit)
{
	quit = Update();
}
```

This while loop will continually loop until the Update function returns true. Whether the function returns true or false will depend on the event checks we add shortly. You should now have in your main an if statement that fires if InitSDL is true and contained in here will be the game loop, and outside the if statement will be a call to CloseSDL.

4. In the Update function body add the following declaration:

```
//Event handler
SDL_Event e;
```

This SDL_Event is local and will go out of scope at the end of the *Update()* function. We will use this variable to store any event that has taken place.

a. Next, we need to poll SDL for any events that have taken place.

```
//get events
SDL_PollEvent(&e);
```

We pass the event variable 'e' by reference (passing the address) so it can be adjusted in the SDL_PollEvent function.

b. Once we have our event, we need to determine what action is to be taken. We will use a switch statement:

```
//handle the events
switch (e.type)
{
	//click the 'X' to quit
case SDL_QUIT:
	return true;
	break;
}

return false;
```

SDL_QUIT is the event type returned when the X is clicked. As the user has decided to close the application, we should return true, which will in turn set the flag in our main function to stop the while loop. We finally wrap the function up with a return false. This will be repeatedly reached provided the user doesn't trigger the 'X' event to close the program.

5. Build and run the program and click the ‘X’ button to ensure it terminates the program.

## Additional Work

As a further example of events and for you to have a play, try and make the screen close on a key press of ‘Q’ or by clicking the right mouse button anywhere.

To determine the key press you will need to use the following format:

```
case SDL_KEYUP:
	switch (e.key.keysym.sym)
```

The following link will give you a complete list of all the keys and there SDL values:
http://www.libsdl.org/release/SDL-1.2.15/docs/html/sdlkey.html

To determine mouse clicks, try this within your switch statement:

```
case SDL_MOUSEBUTTONDOWN:
```

In the next tutorial we will be looking at getting images drawn to the screen.