Handling Keyboard Input with SDL

Note: This tutorial assumes that you already know how to display a window and draw a sprite with SDL.

Checking for Key Presses

When we need to get event information, we pass an `SDL_Event` structure to `SDL_PollEvent()`, which fills the event structure with event information, or returns 0 if there was no event. The type of event that happened is stored in the `SDL_Event` structure's type variable. If type is equal to `SDL_QUIT`, we know that the user wants to close the window.

To see if the user has pressed a key, we check to see if type is equal to `SDL_KEYDOWN`. To find out which key has been pressed, we check the `event.key.keysym.sym` variable, which is an `SDLKey`.

First of all, don't worry about the whole "`event.key.keysym.sym`" thing. That's just where the creators of SDL stored the key information. `sym` is an `SDLKey` variable that is contained in `keysym,` which is contained in key, which is contained in the `SDL_Event` structure. Since this is a bit convoluted, I like to put the information into my own `SDLKey` variable like so:

```
SDLKey keyPressed = event.key.keysym.sym;
```

This saves typing and makes things easier to read.

An `SDLKey` variable can take on a number of values corresponding to the keys on the keyboard. For the next example, we'll check to see if our `SDLKey` variable is equal to `SDLK_ESCAPE`. To see all of the values that an `SDLKey` variable can be, look at "SDL_keysym.h" in the **include** folder in your SDL directory.

The following code displays a window and closes it if the user presses **Esc**. I've bolded the code that checks for key presses.

```
#include "SDL.h"

const int WINDOW_WIDTH = 640;
const int WINDOW_HEIGHT = 480;
const char* WINDOW_TITLE = "SDL Start";
```

```
int main(int argc, char **argv)
{
   SDL_Init( SDL_INIT_VIDEO );
   SDL_Surface* screen = SDL_SetVideoMode( WINDOW_WIDTH,
WINDOW_HEIGHT, 0,
      SDL_HWSURFACE | SDL_DOUBLEBUF );
   SDL_WM_SetCaption( WINDOW_TITLE, 0 );
   SDL_Event event;
   bool gameRunning = true;
   while (gameRunning)
   {
      if (SDL_PollEvent(&event))
      {
         if (event.type == SDL_QUIT)
         {
            gameRunning = false;
         }

         if (event.type == SDL_KEYDOWN)
         {
            SDLKey keyPressed = event.key.keysym.sym;

            switch (keyPressed)
            {
               case SDLK_ESCAPE:
                  gameRunning = false;
                  break;
            }
         }
      }
   }
   SDL_Quit();
   return 0;
}
```

The code above checks for key presses. If you need to check to see if a key had been released, you replace `SDL_KEYDOWN` with `SDL_KEYUP`. The rest will be the same.

Getting a Sprite We Can Move

The normal thing to do with keyboard input is to make something move around when the arrow keys are pressed. Before we can do that though, we need something that we can move around.

The following code draws a sprite at the location stored in the variables batX and batY.

When we want to move the sprite, we just have to change these variables. The bat image is stored in this file. If you want to run the code, be sure to copy it into your project directory.

```
#include "SDL.h"

const int WINDOW_WIDTH = 640;
const int WINDOW_HEIGHT = 480;
const char* WINDOW_TITLE = "SDL Start";

void drawSprite(SDL_Surface* imageSurface,
                SDL_Surface* screenSurface,
                int srcX, int srcY,
                int dstX, int dstY,
                int width, int height);

int main(int argc, char **argv)
{
   SDL_Init( SDL_INIT_VIDEO );

   SDL_Surface* screen = SDL_SetVideoMode( WINDOW_WIDTH,
WINDOW_HEIGHT, 0,
      SDL_HWSURFACE | SDL_DOUBLEBUF );
   SDL_WM_SetCaption( WINDOW_TITLE, 0 );

   SDL_Surface* bitmap = SDL_LoadBMP("bat.bmp");
   SDL_SetColorKey(bitmap, SDL_SRCCOLORKEY, SDL_MapRGB(bitmap-
>format, 255, 0, 255));

   int batImageX = 24;
   int batImageY = 63;
   int batWidth = 65;
   int batHeight = 44;

   // We change these to make the bat move
   int batX = 100;
   int batY = 100;

   SDL_Event event;
   bool gameRunning = true;

   while (gameRunning)
   {
      // Handle input
      if (SDL_PollEvent(&event))
      {
         if (event.type == SDL_QUIT)
         {
            gameRunning = false;
         }

         if (event.type == SDL_KEYDOWN)
         {
```

```
            SDLKey keyPressed = event.key.keysym.sym;

            switch (keyPressed)
            {
               case SDLK_ESCAPE:
                  gameRunning = false;
                  break;
            }
         }
      }

      // Draw the scene
      SDL_FillRect(screen, NULL, SDL_MapRGB(screen->format, 0, 0,
0));

      drawSprite(bitmap,
                 screen,
                 batImageX, batImageY,
                 batX, batY,
                 batWidth, batHeight);

      SDL_Flip(screen);
   }

   SDL_FreeSurface(bitmap);

   SDL_Quit();

   return 0;
}

void drawSprite(SDL_Surface* imageSurface,
                SDL_Surface* screenSurface,
                int srcX, int srcY,
                int dstX, int dstY,
                int width, int height)
{
   SDL_Rect srcRect;
   srcRect.x = srcX;
   srcRect.y = srcY;
   srcRect.w = width;
   srcRect.h = height;

   SDL_Rect dstRect;
   dstRect.x = dstX;
   dstRect.y = dstY;
   dstRect.w = width;
   dstRect.h = height;

   SDL_BlitSurface(imageSurface, &srcRect, screenSurface, &dstRect);
}
```

Moving the Sprite When the User Presses a Key

When an arrow key is pressed, `keyPressed` will be equal to `SDLK_LEFT`, `SDLK_RIGHT`, `SDLK_UP`, or `SDLK_DOWN`.

Here are the changes to the input code's switch statement. Remember that positive y is down.

```
switch (keyPressed)
{
   case SDLK_ESCAPE:
      gameRunning = false;
      break;
   case SDLK_LEFT:
      batX -= 1;
      break;
   case SDLK_RIGHT:
      batX += 1;
      break;
   case SDLK_UP:
      batY -= 1;
      break;
   case SDLK_DOWN:
      batY += 1;
      break;
}
```

Moving the Sprite When the User Holds a Key

If you compiled and ran the code so far, you'll have found that you had to repeatedly tap the arrow keys to make the sprite move. This obviously won't work for most games. We'd much rather have the sprite move constantly when the player holds a key down.

To do this, we'll have an array that stores a boolean value for each key on the keyboard. If the value at a key's index in the array is true, the key is pressed. If it's false, the key is not pressed. We'll set a key's value to true when `event.type` equals `SDL_KEYDOWN` and false when `event.type` equals `SDL_KEYUP`.

Our job is made easy by the fact that the enumerations of the keys on the keyboard range from 0 to 322. All we have to do is create an arary of 323 booleans and index it with the key enumerations (eg. `array[SDLK_LEFT]`).

Each time the game loop runs, we'll update the key array and then check its values to see

if we should move our sprite. Here is the code to do this. I've bolded the relevant code.

```
#include "SDL.h"

const int WINDOW_WIDTH = 640;
const int WINDOW_HEIGHT = 480;
const char* WINDOW_TITLE = "SDL Start";

void drawSprite(SDL_Surface* imageSurface,
                SDL_Surface* screenSurface,
                int srcX, int srcY,
                int dstX, int dstY,
                int width, int height);

int main(int argc, char **argv)
{
   SDL_Init( SDL_INIT_VIDEO );

   SDL_Surface* screen = SDL_SetVideoMode( WINDOW_WIDTH,
WINDOW_HEIGHT, 0,
      SDL_HWSURFACE | SDL_DOUBLEBUF );
   SDL_WM_SetCaption( WINDOW_TITLE, 0 );

   SDL_Surface* bitmap = SDL_LoadBMP("bat.bmp");
   SDL_SetColorKey(bitmap, SDL_SRCCOLORKEY, SDL_MapRGB(bitmap-
>format, 255, 0, 255));

   int batImageX = 24;
   int batImageY = 63;
   int batWidth = 65;
   int batHeight = 44;

   // We change these to make the bat move
   int batX = 100;
   int batY = 100;

   SDL_Event event;
   bool gameRunning = true;

   bool keysHeld[323] = {false}; // everything will be initialized to
false

   while (gameRunning)
   {
      // Handle input
      if (SDL_PollEvent(&event))
      {
         if (event.type == SDL_QUIT)
         {
            gameRunning = false;
         }

         if (event.type == SDL_KEYDOWN)
         {
```

```
            keysHeld[event.key.keysym.sym] = true;
         }
         if (event.type == SDL_KEYUP)
         {
            keysHeld[event.key.keysym.sym] = false;
         }
      }
      if ( keysHeld[SDLK_ESCAPE] )
      {
         gameRunning = false;
      }
      if ( keysHeld[SDLK_LEFT] )
      {
         batX -= 1;
      }
      if ( keysHeld[SDLK_RIGHT] )
      {
         batX += 1;
      }
      if ( keysHeld[SDLK_UP] )
      {
         batY -= 1;
      }
      if (keysHeld[SDLK_DOWN])
      {
         batY += 1;
      }
      // Draw the scene
      SDL_FillRect(screen, NULL, SDL_MapRGB(screen->format, 0, 0,
0));
      drawSprite(bitmap,
                 screen,
                 batImageX, batImageY,
                 batX, batY,
                 batWidth, batHeight);
      SDL_Flip(screen);
   }
   SDL_FreeSurface(bitmap);
   SDL_Quit();
   return 0;
}
void drawSprite(SDL_Surface* imageSurface,
                SDL_Surface* screenSurface,
                int srcX, int srcY,
                int dstX, int dstY,
                int width, int height)
```

```
{
   SDL_Rect srcRect;
   srcRect.x = srcX;
   srcRect.y = srcY;
   srcRect.w = width;
   srcRect.h = height;

   SDL_Rect dstRect;
   dstRect.x = dstX;
   dstRect.y = dstY;
   dstRect.w = width;
   dstRect.h = height;

   SDL_BlitSurface(imageSurface, &srcRect, screenSurface, &dstRect);
}
```

Exercises

Ex 1) Try adding a second sprite that moves when you press the WASD keys.