Getting the Time with SDL

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

The SDL Timer

To initialize SDL's timer, you pass `SDL_INIT_TIMER` to `SDL_Init()`. You can OR this value with `SDL_INIT_VIDEO` like so:

```
SDL_Init( SDL_INIT_VIDEO | SDL_INIT_TIMER );
```

The function `SDL_GetTicks()` will give you the number of milliseconds that have passed since SDL was initialized. If you want this value in seconds, just divide by 1000.

The following code demonstrates how to get the number of seconds that have passed since a certain event.

```
// An event happens
int timeOfEvent = SDL_GetTicks();

// ...
// Pretend a bunch of things happen here
// ...

int timeSinceEvent = (SDL_GetTicks() - timeOfEvent) / 1000;
```

Let's say the event happens after 1200 milliseconds have passed since SDL was initialized. Then let's say that at the point where we do our `timeSinceEvent` calculation, 4200 milliseconds have passed since SDL was initialized. Subtraction yields 4200 - 1200 = 3000 milliseconds and division yields 3000 / 1000 = 3 seconds. Thus we know that 3 seconds have passed since the event happened.

One other time related function I should mention is `SDL_Delay()`. This function takes the number of milliseconds you would like your program to delay for. Your processor will have time to work on other things while you game is delayed.

One thing to keep in mind about `SDL_Delay()` is that you can't depend on it to be any more precise than 10 milliseconds. If you pass it 1, it will most likely delay for 10 milliseconds instead of 1 millisecond. It's not a bad idea to make a call to `SDL_Delay()` at the end of your game loop to give your processor time to do other things.

Using the Timer to Control Your Game's Speed

You always want to control the speed of your game. If you don't, your game will run at different speeds on different computers.

For games, you control the speed of your objects by moving them at a specified rate. For our purposes, we'll move our objects a certain number of pixels every second. We do this with the following formula:

```
x = x + dt * velocityX;
y = y + dt * velocityY;
```

dt stands for "delta time" which is the change in time. Let's say we want an object to move 10 pixels every second and that 4 seconds have passed (dt == 4). We'll want to move the object dt * velocity pixels or, in this case, 4 * 10 = 40 pixels.

Since our game loop will be running many times a second, we can't count on the delta time to be greater than or equal to one. If we store it as an integer, our object won't move because the delta time will always be zero. The same goes for if we store the location or velocity variables as integers. We need to store these values as floats.

Here is some sample code that draws a sprite and moves it around. At the beginning of the game loop, it gets the time delta by subtracting the previous time from the current time. It then sets the previous time variable to the current time for the next time the game loop runs. I've bolded the relevant parts of the code.

```
#include "SDL.h"

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

// Pixels per second
const float BAT_SPEED_X = 100.0;
const float BAT_SPEED_Y = 50.0;

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_INIT_TIMER );

   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
   float batX = 0.0;
   float batY = 100.0;

   // Each frame, we add these to the bat's location variables to
make it move
   float batSpeedX = 0.0;
   float batSpeedY = 0.0;

   float deltaTime = 0.0;
   int thisTime = 0;
   int lastTime = 0;

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

   while (gameRunning)
   {
      thisTime = SDL_GetTicks();
      deltaTime = (float)(thisTime - lastTime) / 1000;
      lastTime = thisTime;

      // 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] )
      {
         batSpeedX = -BAT_SPEED_X;
      }
      if ( keysHeld[SDLK_RIGHT] )
      {
         batSpeedX = BAT_SPEED_X;
      }
      if ( keysHeld[SDLK_UP] )
      {
         batSpeedY = -BAT_SPEED_Y;
      }
      if (keysHeld[SDLK_DOWN])
      {
         batSpeedY = BAT_SPEED_Y;
      }
      // Move the bat
      batX += batSpeedX * deltaTime;
      batY += batSpeedY * deltaTime;
      // 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);
      // Give the computer a break (optional)
      SDL_Delay(1);
   }
   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);
}
```