Lazy Foo' Productions

Frame Independent Movement

Last Updated 11/15/07

Before we had to cap the frame rate to keep the dot from moving too fast. Here we're going to make the dot move based on time instead of frame rate so it moves the same no matter what the frame rate is.

```
//The attributes of the dot
const int DOT_WIDTH = 20;
const int DOT_HEIGHT = 20;
const int DOT_VEL = 200;
```

Here we define the dot related constants of width, height, and velocity.

Now that we're moving in relation to time instead of frame rate we have to change how we handle velocity. Instead of basing velocity per frame we're going to base it per second. In the original motion tutorial we had the dot travel 10 pixels every frame. Since the program ran at 20 frames per second that meant the dot moved at a rate of 200 pixels per second.

So now the dot's velocity is 200 pixels per second.

```
//The dot
class Dot
{
    private:
    //The X and Y offsets of the dot
    float x, y;

    //The velocity of the dot
    float xVel, yVel;

    public:
    //Initializes the variables
```

```
    Dot();

    //Takes key presses and adjusts the dot's velocity
    void handle_input();

    //Moves the dot
    void move( Uint32 deltaTicks );

    //Shows the dot on the screen
    void show();
};
```

Here we have yet another revision of our friend the dot class.

Now the offsets and velocity are floating point numbers. This is because there will be cases where the dot will move less than a pixel per frame.

Say this program runs at 300 fps on a computer. To move at 200 pps the dot would have to move at 2/3 of a pixel per frame.

We also have a move() function that takes in a delta time. For those of you who haven't taken physics, a delta time is a change in time. We need the change in time since the last frame to determine how much the dot needs to move.

And of course we have our video and input functions.

```
void Dot::handle_input()
{
    //If a key was pressed
    if( event.type == SDL_KEYDOWN )
    {
        //Adjust the velocity
        switch( event.key.keysym.sym )
        {
            case SDLK_UP: yVel -= DOT_VEL; break;
            case SDLK_DOWN: yVel += DOT_VEL; break;
            case SDLK_LEFT: xVel -= DOT_VEL; break;
            case SDLK_RIGHT: xVel += DOT_VEL; break;
        }
    }
    //If a key was released
    else if( event.type == SDL_KEYUP )
    {
        //Adjust the velocity
        switch( event.key.keysym.sym )
        {
            case SDLK_UP: yVel += DOT_VEL; break;
            case SDLK_DOWN: yVel -= DOT_VEL; break;
            case SDLK_LEFT: xVel += DOT_VEL; break;
            case SDLK_RIGHT: xVel -= DOT_VEL; break;
        }
    }
}
```

As you can see the input handling is pretty much the same from last time.

```
void Dot::move( Uint32 deltaTicks )
{
    //Move the dot left or right
    x += xVel * ( deltaTicks / 1000.f );

    //If the dot went too far to the left
    if( x < 0 )
```

```
    {
        //Move back
        x = 0;
    }
    //or the right
    else if( x + DOT_WIDTH > SCREEN_WIDTH )
    {
        //Move back
        x = SCREEN_WIDTH - DOT_WIDTH;
    }

    //Move the dot up or down
    y += yVel * ( deltaTicks / 1000.f );

    //If the dot went too far up
    if( y < 0 )
    {
        //Move back
        y = 0;
    }
    //or down
    else if( y + DOT_HEIGHT > SCREEN_HEIGHT )
    {
        //Move back
        y = SCREEN_HEIGHT - DOT_HEIGHT;
    }
}
```

Here is where we do our movement.

We take in the delta time which is going tell us the change in time since the dot last moved. If the program is running at 100 fps, the delta time will be 1/100 of a second. If the program is running at 200 fps, the delta time will be 1/200 of a second. If the program is running at 150 fps, the delta time will be 1/150 of a second, so on and so on.

The formula to figure out how much we need to move is:

velocity in pixels per second * time since last frame in seconds.

So if the program runs at 200 frames per second:

200 pps * 1/200 seconds = 1 pixel

If the program runs at 100 frames per second:

200 pps * 1/100 seconds = 2 pixels

and so on and so on.

Using the time based movement makes sure that the dot always moves at 200 pps.

Also notice we changed our method to keep the dot in bounds. Instead of using the undo motion method like before, here whenever the dot goes off the screen we pull it back in.

```
void Dot::show()
{
    //Show the dot
    apply_surface( (int)x, (int)y, dot, screen );
}
```

Here you see the show() function is pretty much unchanged except for the fact that we do have to type cast the float offsets to integers for blitting.

```
    //Quit flag
    bool quit = false;

    //The dot that will be used
    Dot myDot;

    //Keeps track of time since last rendering
    Timer delta;

    //Initialize
    if( init() == false )
    {
        return 1;
    }

    //Load the files
    if( load_files() == false )
    {
        return 1;
    }

    //Start delta timer
    delta.start();
```

Here's the top of our main() function.

Along with our dot we make a timer object to measure the delta time between frames. We start the timer before we enter our main loop.

```
    //While the user hasn't quit
    while( quit == false )
    {
        //While there's events to handle
        while( SDL_PollEvent( &event ) )
        {
            //Handle events for the dot
            myDot.handle_input();

            //If the user has Xed out the window
            if( event.type == SDL_QUIT )
            {
                //Quit the program
                quit = true;
            }
        }

        //Move the dot
        myDot.move( delta.get_ticks() );

        //Restart delta timer
        delta.start();
```

At the top of our main loop we handle events and move the dot.

After the dot is moved we restart the delta timer so we can keep track of how long it's been since we last moved.

```
        //Fill the screen white
        SDL_FillRect( screen, &screen->clip_rect, SDL_MapRGB( screen->format, 0xFF, 0xFF, 0xFF )

        //Show the dot on the screen
        myDot.show();
```

```
        //Update the screen
        if( SDL_Flip( screen ) == -1 )
        {
            return 1;
        }
    }
```

Then we we do our graphics as we do normally.

As you can see we don't cap the frame rate, but since our movement is based on time it doesn't matter what the frame rate is.

Download the media and source code for this tutorial here.