**Frame Rate in GameMaker**

Hi there, Sanbox here. I’m the producer of **obj\_podcast** (mostly making sure Seb wakes up for the podcast). I’m here to talk to you about **mastering the power of time**, except only in video games. Almost as good.

Spend any significant amount of time in any GameMaker (or other engines) community, and some fancy guy with a 144hz monitor will ask “how do I make my game frame independent?” to which a chorus of intelligent, brave, and extraordinarily handsome Linux users, beard and all, will respond “Delta time”. So that’s where we start.

**Delta Time Won’t Save You (at least, not on its own)**

Let’s go through a very simple example:

```  
#create

speed = 5;  
  
#step  
x += speed;  
```  
This is not framerate dependent. So let’s do this:  
```  
#step\_of\_some\_manager

global.dt = delta\_time/1000000\*60;  
```

Here, we create a global called **dt** which, when **delta\_time** is 1/60th of a second (in microseconds, thus the big division to convert to second) will be **1**. Crucially, at 30 FPS, this number will be 2, and at 120 FPS, this number will be 0.5. Therefore, speeds will be twice as “fast” at 30 FPS (but the whole game will be running half as fast), and half as “fast” at 120 FPS (but the whole game will be running twice as fast).

Now, let’s apply that to our simple example:

```  
#create

speed = 5;  
  
#step  
x += speed \* global.dt;  
```

Okay, so now let’s plot. Assuming our X begins at 0, we’ll see the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Time** | **30 FPS** | **60 FPS** | **120 FPS** |
| **0.00000** | 0 | 0 | 0 |
| **0.00833** | - | - | 2.5 |
| **0.01667** | - | 5 | 5 |
| **0.02500** | - | - | 7.5 |
| **0.03333** | 10 | 10 | 10 |

Looking good! Those `-` represent frames where one FPS isn’t being ran. As you see, 120 FPS, being 4x 30 FPS, will have ran 4 frames before 30 FPS will run even on.

If this is your game, you’re good to go!

Unfortunately…this isn’t your game. Let’s transition to doing something a little more realistic: jumping.

Here’s our naïve implementation (the simplest way to program it, essentially):

```  
#create

grav = 2;

veloc = -20;

y += veloc;

#step

veloc += accel;

veloc = min(veloc, max\_veloc) \* global.dt;

x += veloc;  
```

I’m using physics terms here, but this is the setup that Heartbeast, Shaun Spalding, and many others teach as the basic movement.

Let’s map this out, applying our delta time like we did before, but let’s just go straight to moment 16.667, which is the 1000th frame at 60 FPS (2000th at 120, 500th at 30, etc):

|  |  |  |  |
| --- | --- | --- | --- |
| **Time** | **30 FPS** | **60 FPS** | **120 FPS** |
| **16.66667** | 4985 | 4992.5 | 4995 |

Well hey, that’s not too bad. We’re off by about 2.5 pixels at 120, and we’re off by a bit more the other way. But those aren’t too bad.

Now, unfortunately, I have some bad news to deliver. First, monitors don’t come in at 120hz. They typically are at 144hz.