How do you separate game logic from display?

Asked 16 years, 4 months ago Modified 5 years, 1 month ago

Viewed 12k times

20

How can you make the display frames per second be independent from the game logic? That is so the game logic runs the same speed no matter how fast the video card can render.

algorithm

Share

Improve this question

Follow

edited Aug 24, 2008 at 15:39

Chris

6,842 ● 6 ● 53 ● 67

asked Aug 20, 2008 at 4:30

azaziel

8 Answers

Sorted by: Highest score (default)

32

I think the question reveals a bit of misunderstanding of how game engines should be designed. Which is perfectly ok, because they are damn complex things that are difficult to get right ;)



You are under the correct impression that you want what is called Frame Rate Independence. But this does not only refer to Rendering Frames.

A Frame in single threaded game engines is commonly referred to as a Tick. Every Tick you process input, process game logic, and render a frame based off of the results of the processing.

What you want to do is be able to process your game logic at any FPS (Frames Per Second) and have a deterministic result.

This becomes a problem in the following case:

Check input: - Input is key: 'W' which means we move the player character forward 10 units:

```
playerPosition += 10;
```

Now since you are doing this every frame, if you are running at 30 FPS you will move 300 units per second.

But if you are instead running at 10 FPS, you will only move 100 units per second. And thus your game logic is *not* Frame Rate Independent.

Happily, to solve this problem and make your game play logic Frame Rate Independent is a rather simple task.

First, you need a timer which will count the time each frame takes to render. This number in terms of seconds

(so 0.001 seconds to complete a Tick) is then multiplied by what ever it is that you want to be Frame Rate Independent. So in this case:

When holding 'W'

```
playerPosition += 10 * frameTimeDelta;
```

(Delta is a fancy word for "Change In Something")

So your player will move some fraction of 10 in a single Tick, and after a full second of Ticks, you will have moved the full 10 units.

However, this will fall down when it comes to properties where the rate of change also changes over time, for example an accelerating vehicle. This can be resolved by using a more advanced integrator, such as "Verlet".

Multithreaded Approach

If you are still interested in an answer to your question (since I didn't answer it but presented an alternative), here it is. Separating Game Logic and Rendering into different threads. It has it's draw backs though. Enough so that the vast majority of Game Engines remain single threaded.

That's not to say there is only ever one thread running in so called single threaded engines. But all significant tasks are usually in one central thread. Some things like

Collision Detection may be multithreaded, but generally the Collision phase of a Tick blocks until all the threads have returned, and the engine is back to a single thread of execution.

Multithreading presents a whole, very large class of issues, even some performance ones since everything, even containers, must be thread safe. And Game Engines are very complex programs to begin with, so it is rarely worth the added complication of multithreading them.

Fixed Time Step Approach

Lastly, as another commenter noted, having a Fixed size time step, and controlling how often you "step" the game logic can also be a very effective way of handling this with many benefits.

Linked here for completeness, but the other commenter also links to it: [Fix Your Time Step]

Share Improve this answer

Follow

edited May 31, 2012 at 19:29

Community Bot

1 • 1

answered Aug 20, 2008 at 4:41

Adam

26.5k • 23 • 77 • 87

Can you please elaborate how to achieve time synchronization over more devices (multiplayer games)?

Since for each device single-threaded approach would take different amount of time... if you have some materials on that I could read, I would be most grateful – arenaq May 21, 2015 at 11:55

That is a much much more difficult topic I couldn't begin to answer here. I suggest googling about it, there are lots of good resources out there. – Adam May 21, 2015 at 18:23

8

Koen Witters has a very detailed article about different game loop setups.

He covers:

- FPS dependent on Constant Game Speed
- Game Speed dependent on Variable FPS
- Constant Game Speed with Maximum FPS
- Constant Game Speed independent of Variable FPS

(These are the headings pulled from the article, in order of desirability.)

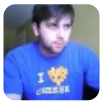
Share Improve this answer Follow

edited Oct 26, 2019 at 10:12

Xiazee

3 ● 3

answered Aug 20, 2008 at 4:52



Ryan Fox

10.2k ● 5 ● 39 ● 48

You could make your game loop look like:

4

```
int lastTime = GetCurrentTime();
while(1) {
    // how long is it since we last updated?
    int currentTime = GetCurrentTime();
    int dt = currentTime - lastTime;
    lastTime = currentTime;

    // now do the game logic
    Update(dt);

    // and you can render
    Draw();
}
```

Then you just have to write your `Update()` function to take into account the time differential; e.g., if you've got an object moving at some speed `v`, then update its position by `v * dt` every frame.

Share Improve this answer

Follow

answered Aug 20, 2008 at 4:37

Jesse Beder

34k ● 22 ● 110 ● 146

2

There was an excellent article on flipcode about this back in the day. I would like to dig it up and present it for you.

http://www.flipcode.com/archives/Main_Loop_with_Fixed_Time_Steps.shtml

It's a nicely thought out loop for running a game:

1. Single threaded
2. At a fixed game clock

3. With graphics as fast as possible using an interpolated clock

Well, at least that's what I think it is. :-) Too bad the discussion that pursued after this posting is harder to find. Perhaps the wayback machine can help there.

```
time0 = getTickCount();
do
{
  time1 = getTickCount();
  frameTime = 0;
  int numLoops = 0;

  while ((time1 - time0)  TICK_TIME && numLoops <
MAX_LOOPS)
  {
    GameTickRun();
    time0 += TICK_TIME;
    frameTime += TICK_TIME;
    numLoops++;
// Could this be a good idea? We're not doing it,
anyway.
//    time1 = getTickCount();
  }
  IndependentTickRun(frameTime);

  // If playing solo and game logic takes way too
long, discard pending
time.
  if (!bNetworkGame && (time1 - time0)  TICK_TIME)
    time0 = time1 - TICK_TIME;

  if (canRender)
  {
    // Account for numLoops overflow causing
percent  1.
    float percentWithinTick = Min(1.f, float(time1
- time0)/TICK_TIME);
    GameDrawWithInterpolation(percentWithinTick);
  }
```

```
}
while (!bGameDone);
```

Share Improve this answer

Follow

answered Aug 20, 2008 at 5:06

Frank Krueger

70.9k ● 48 ● 164 ● 211

0

[Enginuity] has a slightly different, but interesting approach: the Task Pool.

Share Improve this answer

Follow

answered Aug 21, 2008 at 13:33

GhassanPL

2,724 ● 5 ● 32 ● 40

0

Single-threaded solutions with time delays before displaying graphics are fine, but I think the progressive way is to run game logic in one thread, and displaying in other thread.

But you should synchronize threads right way ;) It'll take a long time to implement, so if your game is not too big, single-threaded solution will be fine.

Also, extracting GUI into separate thread seems to be great approach. Have you ever seen "Mission complete" pop-up message while units are moving around in RTS games? That's what I'm talking about :)

Share Improve this answer

Follow

answered Sep 15, 2008 at 14:26

realsugar

412 ● 2 ● 8

0

This doesn' cover the higher program abstraction stuff, i.e. state machines etc.

It's fine to control movement and acceleration by adjusting those with your frame time lapse. But how about stuff like triggering a sound 2.55 seconds after this or that, or changing game level 18.25 secs later, etc.

That can be tied up to an elapsed frame time accumulator (counter), BUT these timings can get screwed up if your frame rate falls below your state script resolution i.e if your higher logic needs 0.05 sec granularity and you fall below 20fps.

Determinism can be kept if the game logic is run on a separate "thread" (at the software level, which I would prefer for this, or OS level) with a fixed time-slice, independent of fps.

The penalty might be that you might waste cpu time in-between frames if not much is happening, but I think it's probably worth it.

Share Improve this answer

Follow

answered Jan 29, 2010 at 17:35

Camilo

1

-2

From my experience (not much) Jesse and Adam's answers should put you on the right track.

If you are after further information and insight into how this works, i found that the sample applications for TrueVision 3D were very useful.

Share Improve this answer

Follow

answered Aug 20, 2008 at 4:45

Jarod Elliott

15.7k ● 3 ● 36 ● 34