No blocking user-input & adapt to passage of time

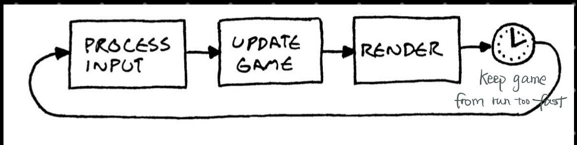
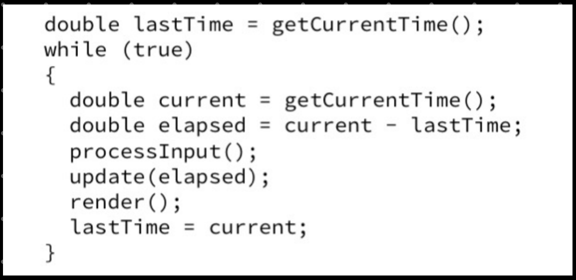


Figure 9.1 – A dead simple game loop


```
double lastTime = getCurrentTime();
while (true)
{
  double current = getCurrentTime();
  double elapsed = current - lastTime;
  processInput();
  update(elapsed);
  render();
  lastTime = current;
}
```

△ when one side goes faster than another things like **floating point error** get accummulated at different rates, hence causes mismatches.

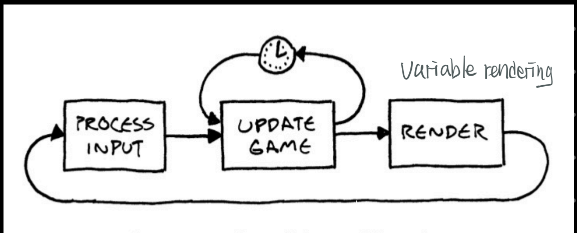
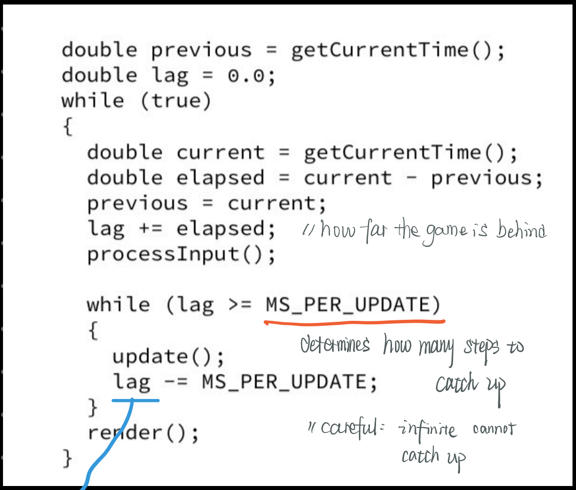


Figure 9.2 – Cutting rendering out of the core loop


```
double previous = getCurrentTime();
double lag = 0.0;
while (true)
{
  double current = getCurrentTime();
  double elapsed = current - previous;
  previous = current;
  lag += elapsed;   // how far the game is behind
  processInput();

  while (lag >= MS_PER_UPDATE)
  {                        Determines how many steps to
    update();              catch up
    lag -= MS_PER_UPDATE;
  }                        // careful: infinite cannot
  render();                catch up
}
```

△ Time step does not have visible **frame rate**

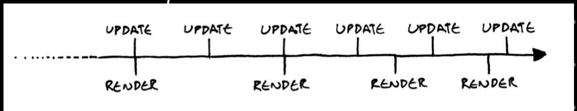
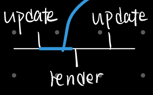


Figure 9.3 – A timeline showing when the game updates and renders

render **whenever we can** → not Steady

not zero leftover



makes motion looks jagged / stuttery

render(lag / MS_PER_UPDATE)
// render Interpolate / guess the motion)
but might be wrong (hit wall actually)