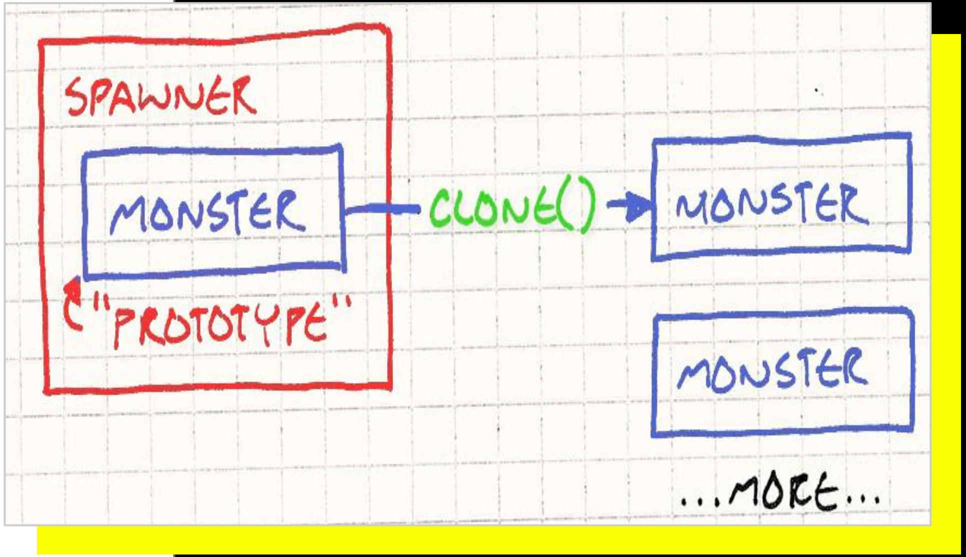
Game Programming Patterns

The Game Loop

Game Programming Patterns

- **Game Loop**
- Update Method
- Component
- Command
- State
- Prototype
- Observer
- ...



x2

What we Want to Avoid!

Flying Spaghetti-Code Monster

Game Programming Patterns

Game Loop Pattern

- **Quintessential** of **Game** Programming **Patterns**
- Almost **every game has** one
- **No two** implementations are exactly **alike**
- Relatively **few programs outside** of **games** use them

Loop Pattern Objective

"Decouple the progression of game time from user input and processor speed."

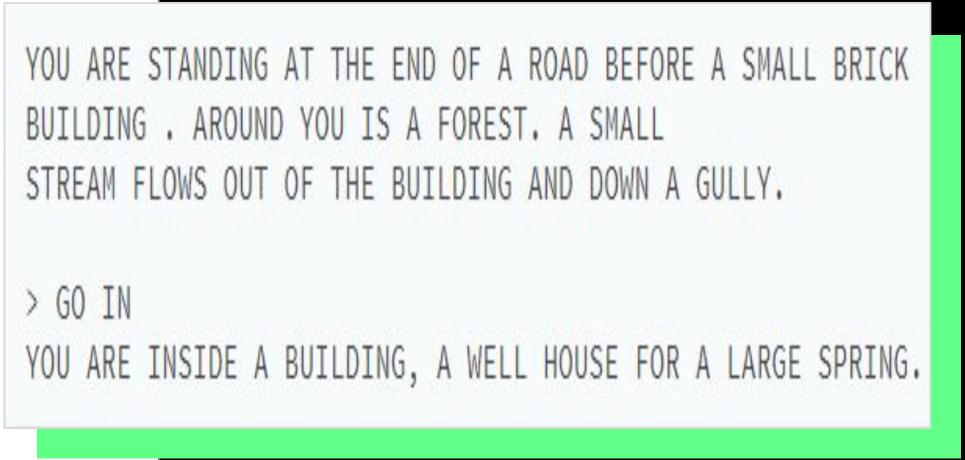
Robert Nystrom

x4

Game Loop: First CLI Programs

CLI Programs

```
while (true)
{
  char* command = readCommand();
  handleCommand(command);
}
```



```
YOU ARE STANDING AT THE END OF A ROAD BEFORE A SMALL BRICK
BUILDING . AROUND YOU IS A FOREST. A SMALL
STREAM FLOWS OUT OF THE BUILDING AND DOWN A GULLY.

> GO IN
YOU ARE INSIDE A BUILDING, A WELL HOUSE FOR A LARGE SPRING.
```

From: Robert Nystrom; “Game Programming Patterns”

Game Loop: Event Loops

Graphic UI Applications

```
while (true)
{
  Event* event = waitForEvent();
  dispatchEvent(event);
}
```

Games

The game loop **processes** user **input**, but **doesn't wait for it**. The loop keeps spinning.

```
while (true)
{
  processInput();
  update();
  render();
}
```

Game Loop: Seconds per second

Non-fixed seconds factor

- Most of the times developers **don't know where** the **game** will be **running**
- This is the **other key job** of a game loop: *"It runs the game at a consistent speed despite differences in the underlying hardware."*

Robert Nystrom

From Chrono Trigger (SNES) intro.

The Game Loop

Keep in Mind

- The **most important code** in a game
- **May need to be coordinated** with the **platform's event loop**
- **Runs** the game **at a consistent speed despite differences** in the underlying **hardware**.

The Pattern

A game loop runs continuously during gameplay. Each turn of the loop, it processes user input without blocking, updates the game state, and renders the game. It tracks the passage of time to control the rate of gameplay.

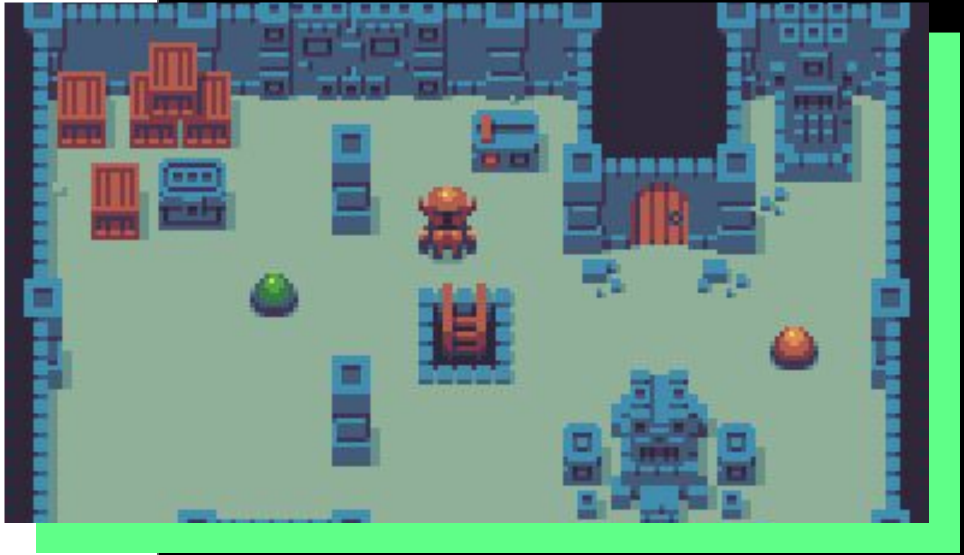
Robert Nystrom

Game Loop: Sample Codes

Naive Implementation

```
while (true)
{
  processInput();
  update();
  render();
}
```

What’s wrong with this implementation?



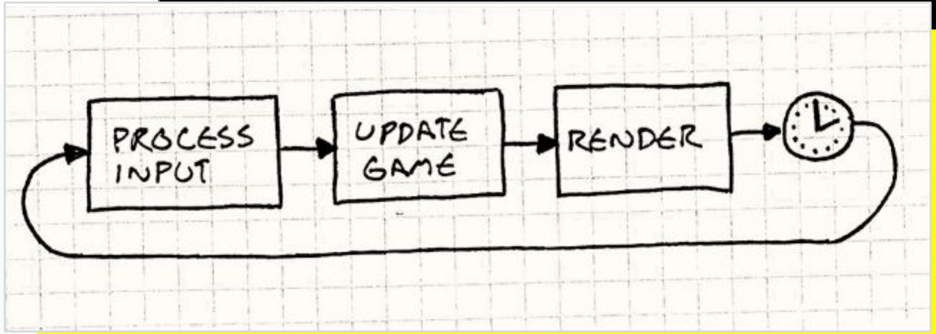
Game Loop: Sample Codes

First Variation

=> Prevent game from running too fast.

```
while (true)
{
  double start = getCurrentTime();
  processInput();
  update();
  render();

  sleep(start + MS_PER_FRAME - getCurrentTime());
}
```

The problem with this implementation?



From: Robert Nystrom; "Game Programming Patterns"

Game Loop: Sample Codes

A Small Improvement

=> **Variable time step:** advance the logic with more frequency

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

The problem with this implementation?

A serious problem is lurking

We've made the game **non-deterministic** and **unstable**.

In order to run in real time, game physics engines are approximations of the real laws of mechanics. To keep those approximations from blowing up, damping is applied.
That damping is carefully tuned to a certain time step. Vary that, and the physics gets unstable.

Robert Nystrom

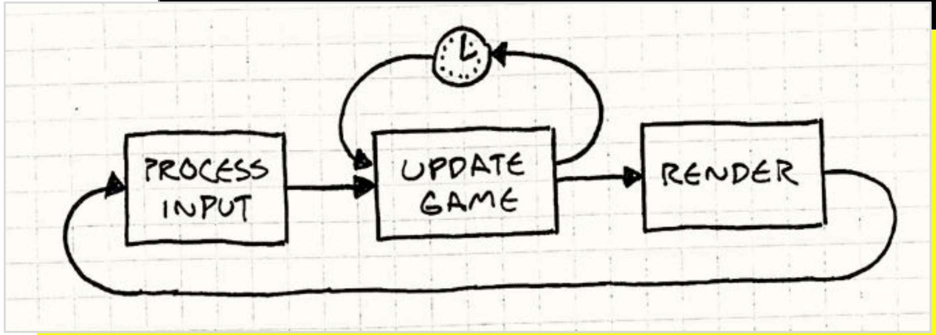
Game Loop: Sample Codes

Solution: Playing Catch Up

```
double previous = getCurrentTime();
double lag = 0.0;
while (true)
{
  double current = getCurrentTime();
  double elapsed = current - previous;
  previous = current;
  lag += elapsed;

  processInput();

  while (lag >= MS_PER_UPDATE)
  {
    update();
    lag -= MS_PER_UPDATE;
  }

  render();
}
```

Fixed time step, variable rendering

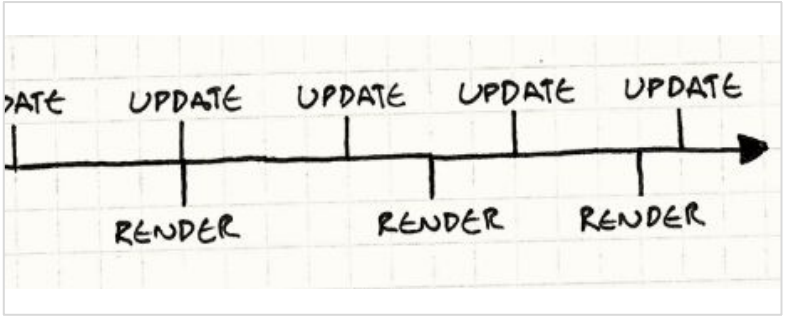
Update the game with a **fixed time step**, but allow **flexibility** on **when to render**.

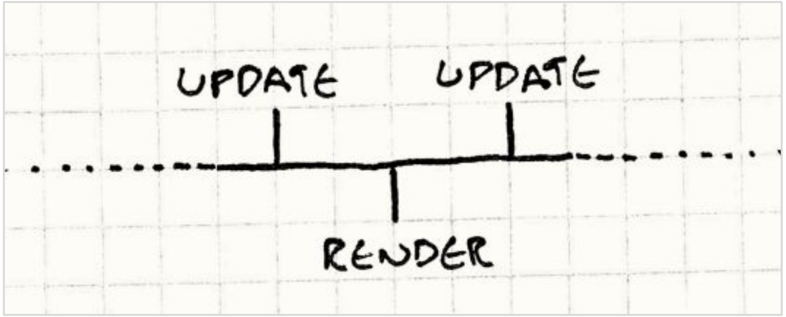


From: Robert Nystrom; "Game Programming Patterns"

Game Loop: Playing Catch Up

One last issue!





Update timeline

- **Update** at a **fixed interval**.
- **Render whenever** we can. Less frequent than updating, but that is ok.
- **Problem:** We don't always render right at the point of updating.

Stuck in the middle

Since the renderer knows each game object and its current velocity.

Solution:

```
render(lag / MS_PER_UPDATE);
```

Game Loop Pattern

Keep in Mind

- **Quintessential** of **Game** Programming **Patterns**
- Almost **every game has** one
- **No two** implementations are exactly **alike**
- Relatively **few programs outside** of **games** use them
- The **most important code** in a game
- **May need to be coordinated** with the **platform's event loop**

Loop Pattern Objective

"Decouple the progression of game time from user input and processor speed."

Robert Nystrom

Game Programming Patterns
The Game Loop