How is a real-world simulation designed?

Asked 16 years, 2 months ago Modified 16 years, 1 month ago

Viewed 1k times

12

I am fascinated by the performance of applications such as "Rollercoaster Tycoon" and "The Sims" and FPS games. I would like to know more about the basic application architecture. (Not so concerned with the UI - I assume MVC/MVP piriciples apply here. Nor am I concerned with the math and physics at this point.)

My main question deals with the tens or hundreds of individual objects in the simulation (people, vehicles, items, etc.) that all move, make decisions, and raise & respond to events - seeming all a the same time, and how they are designed for such good performance.

Q: Primarily, are these objects being processed in a giant loop, one at a time - or is each object processing in it's own thread? How many threads are practical in a simulation like this? (Ballpark figure of course, 10, 100, 1000)

I'm not looking to write a game, I just want the design theory because I'm wondering if such design can apply to other applications where several decisions are being made seemingly at the same time.

multithreading oop

Share

Improve this question

Follow

asked Sep 26, 2008 at 4:17

Doug L.

2,716 ● 1 ● 19 ● 34

5 Answers

Sorted by: Highest score (default)

8

There are two basic ways of doing this kind of simulation Agent Based and System Dynamics. In and agent based simulation each entity in the game would be represented by an instance of a class with properties and behaviors, all the interactions between the entities would have to be explicitly defined and when you want these entities to interact a function gets called the properties of the interacting entities gets changed.

System Dynamics is completely different, it only deals with sums and totals, there is no representation of a single entity in the system. The easiest example of that is the Predator and Prey model.

Both of these have advantages and disadvantages, the System Dynamics approach scales better to large number of entitities while keeping runtime short. While there are multiple formulas that you have to calculate, the time to calculate is independent of the values in the formula. But there is no way to look at an individual entity in this approach. The Agent based approach lets you put

entities in specific locations and lets you interact with specific entities in your simulation.

FSMs and Celular automata are other ways in how to simulate systems in a game. E.g. in the agent based approach you might model the behavior of one agent with a FSM. Simcity used Celular automata to do some of the simulation work.

In general you will probably not have one big huge model that does everything but multiple systems that do specific tasks, some of these will not need to be updated very often e.g. something that determines the weather, others might need constant updates. Even if you put them in separate threads you will want to pause or start them when you need them. You might want to split work over multiple frames, e.g. calculate only updates on a certain number of agents.

Share Improve this answer

Follow

answered Sep 26, 2008 at 10:14

Harald Scheirich

9,754 ● 33 ● 56

4

The source code to the original Simcity has been open sourced as Micropolis. It might be an interesting study.

Share Improve this answer

Follow

answered Oct 31, 2008 at 21:38

postfuturist

22.6k ● 11 ● 66 ● 85

Until very recently, the game's logic and management was in a single thread in a big finite state machine. Now, though, you tend to see the different pieces of the game (audio, graphics, physics, 'simulation' logic, etc) being split into their own FSMs in threads.

Edit: Btw, threads are a very bad way of having things in a simulation happening at the 'same time' -- it leads to race conditions. It's common that when you want to have things going on at the 'same time', you simply figure out what needs to happen as you iterate over your data and store it separately, then apply it once all the data is processed. Rince, repeat.

Share Improve this answer

Follow

answered Sep 26, 2008 at 4:19

Serafina Brocious

30.6k ● 12 ● 91 ● 115

Could each object in it's own thread set its state (properties) "at the same time" and then have a master process deal with the results. No events or races. But the loop doesn't wait as long for each object to process...? – Doug L. Sep 26, 2008 at 4:26

1 You could do this via threadpools or some such fairly efficiently, but I've found it's better to split your data up into blocks and then thread out handlers for this, building up a changeset like I described. – Serafina Brocious Sep 26, 2008 at 4:36

Totally disagree with your answer. If it was as simple as "one fat FSM with bits on different threads" then everyone would be releasing games like this. There is way more to it than this oversimplified response. Have you ever worked on a serious game title before? – OJ. Sep 26, 2008 at 9:22

0

@Cody Brocious

This CodeProject uses Linq to demonstrate this practice. (Linq to Life)

Share Improve this answer

Follow

answered Sep 26, 2008 at 4:34

Nescio

28.4k ● 10 ● 55 ● 76

I had added a comment about how this method wasn't as scalable and such, before I realized that this is just an implementation of it -- damn my lack of LINQ understanding ;) Good link. – Serafina Brocious Sep 26, 2008 at 4:39

0

In addition to the suggestions posted I would recommend browsing the simulation tag at sourceforge. There are a variety of simulation project at varying levels of complexity.

Sourceforge

Also I recommend the following book for a basic overview, While it is focused on physics it deals with issues of simulation.

[Physics for Game Developers]

Share Improve this answer

Follow

answered Sep 26, 2008 at 13:40

RS Conley

7,206 ● 2 ● 22 ● 37

Thank you! I will check out both of these sources. – Doug L.
Sep 26, 2008 at 19:25