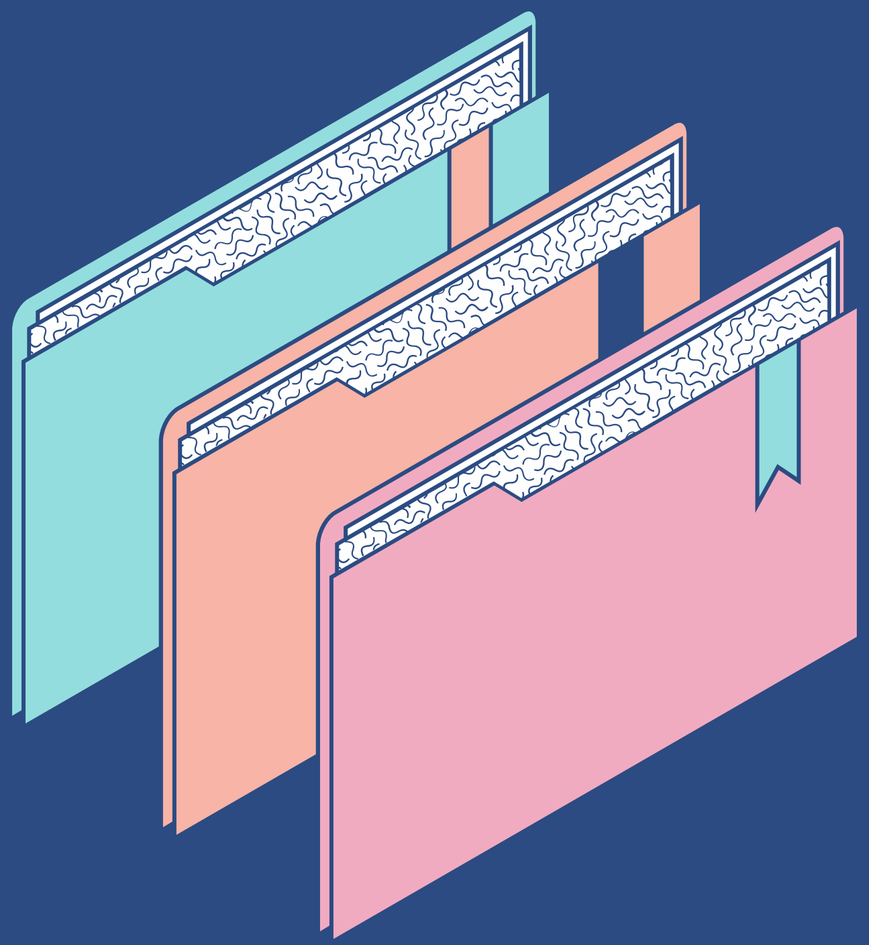
ERTA TERSHANA

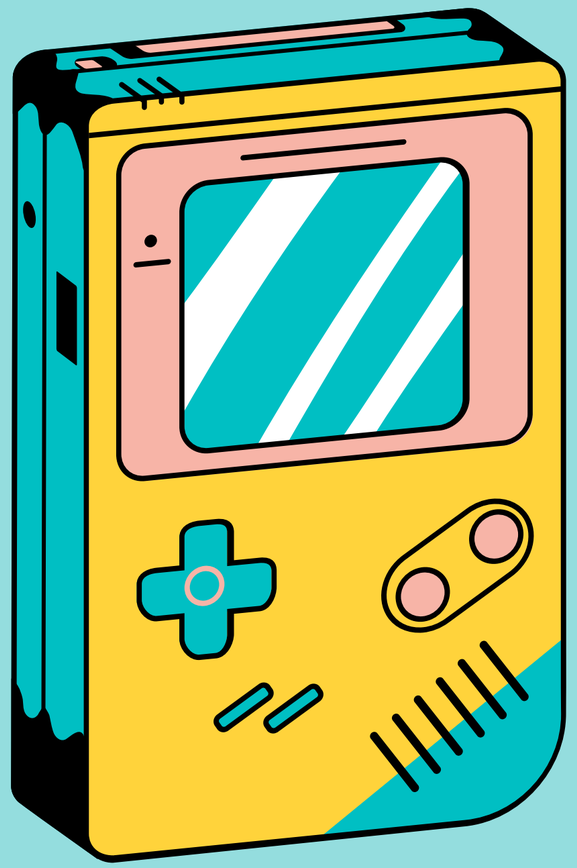
Implementation of parallel programming in games

A look of machine learning that use parallel programming and how it can be implemented on games



Key topics discussed in this presentation

1. Why use parallel programming in games?
2. Why use parallel programming in machine learning?
3. Where can we integrate machine learning into games?
4. A simple example from daily games
5. An example made by me and the commenting of the code



1.Why use parallel programming in games?

Threads are mostly used for island calculations like sounds, physics, particle systems, I/O, path-finding, etc, stuff that can be easily outsourced from the main thread without running into too many locking problems. Yet the main game entity update chain is very hard to distribute over multiple threads with current state-of-the-art engine designs.

2.Why use parallel programming in machine learning?

Distributed processing, a typical ML algorithm involves doing a lot of computation (work/tasks) on a lot of data set. Traditionally, machine learning has been executed in single processor environments, where algorithmic bottlenecks can lead to substantial delays in model processing, from training to classification to distance and error calculation and beyond.

In order to achieve economy of performance. The lack of parallel processing or parallel execution on a shared memory architecture involves doing a lot of utilization(work/tasks) on a lot of data set after performing independent tasks. So in my opinion parallel programming is hard.

3.Integration of machine learning in games

1

Algorithms Playing as NPCs

NPCs will respond to actions in unique and unexpected ways.

2

Modeling Complex Systems

The game can predict and alter downstream effects.

3

Making Games More Beautiful

Textures and objects will render automatically as you get closer to them.

4

More Realistic Interactions

NPCs will create more realistic conversations and responses.

5

Universe Creation on the Fly

Open world games have the potential of having an infinite size.

6

More Engaging Mobile Games

Ai chips in phones can bring the power of ML to phones.

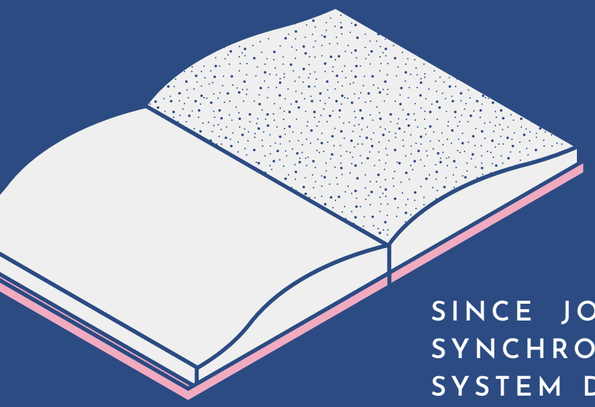
4.A simple example from daily games

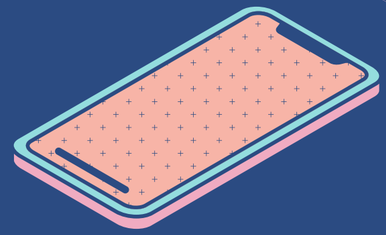
DOOM 3 BFG EDITION

This game is heavily multi-threaded. On PC the game starts with three threads:

1. Thread dedicated to the Renderer Backend (Sends draw commands to GPU).
2. Thread dedicated to Game Logic and Renderer Frontend (Generates draw commands).
3. Thread dedicated to High frequency (250Hz) joystick sampling.

Continuation of 4



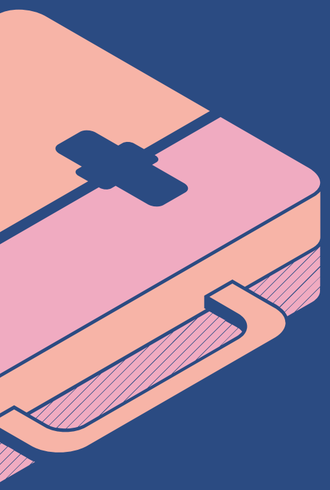


SINCE JOBS ARE SEGREGATED INTO SECTIONS ACCESSED BY ONLY ONE THREAD, THERE IS NO SYNCHRONIZATION REQUIRED FOR ADDING A JOB. HOWEVER, SUBMITTING A JOB TO THE WORKER SYSTEM DOES INVOLVE A MUTEX. HERE IS A EXAMPLE WHERE THE RENDERER TRIES TO FIND WHICH LIGHTS ARE GENERATING INTERACTIONS.

```
//TR.FRONTENDJOBLIST IS A IDPARALLELJOBLIST OBJECT.

FOR ( VIEWLIGHT_T * VLIGHT = TR.VIEWDEF->VIEWLIGHTS; VLIGHT != NULL; VLIGHT = VLIGHT->NEXT )
{
   TR.FRONTENDJOBLIST->ADDJOB( (JOBRUN_T)R_ADDSINGLELIGHT, VLIGHT );
}

TR.FRONTENDJOBLIST->SUBMIT();
TR.FRONTENDJOBLIST->WAIT();
```

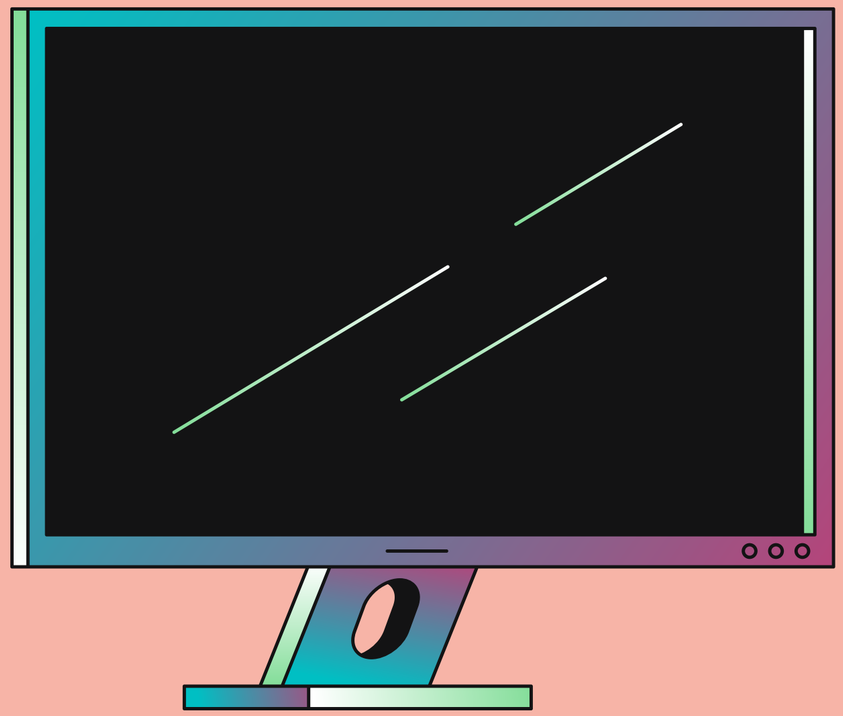


THREE PARTS:
ADDJOB : NO SYNCHRONIZATION NECESSARY, JOB IS ADDED TO A VECTOR.

SUBMIT : MUTEX SYNCHONIZATION, EACH WORKER THREADS ADD THE JOBLIST TO THEIR OWN LOCAL RINGER BUFFER LIST OF JOBLISTS.

WAIT : SIGNAL SYNCHONIZATION (DELEGATED TO OS). LET THE WORKER THREADS COMPLETE.

5.An example made by me and the commenting of the code



The game that I have managed to re-create is Space invaders- 1978 created developed by Tomohiro Nishikado. The game is quite simple and does not use threads, since the software it self can parallize the work without a problem.
I have created two simple functions, one that plays the music in the background and one that plays sound when the player shoots. This is done without interrupting any game setting, which make it a parallel working.

Do you have any questions?

Send it to me! I hope you learned something new.