Cpp North 2025

Taming a Beast

Using ONNX Runtime in AAA Games

Jean-Simon Lapointe

I'm Jean-Simon Lapointe (He/Him) (J-S)

Tech. Lead in Production Technologies at Ubisoft

Using C++ daily for almost 25 years

Working in Computer vision and image processing

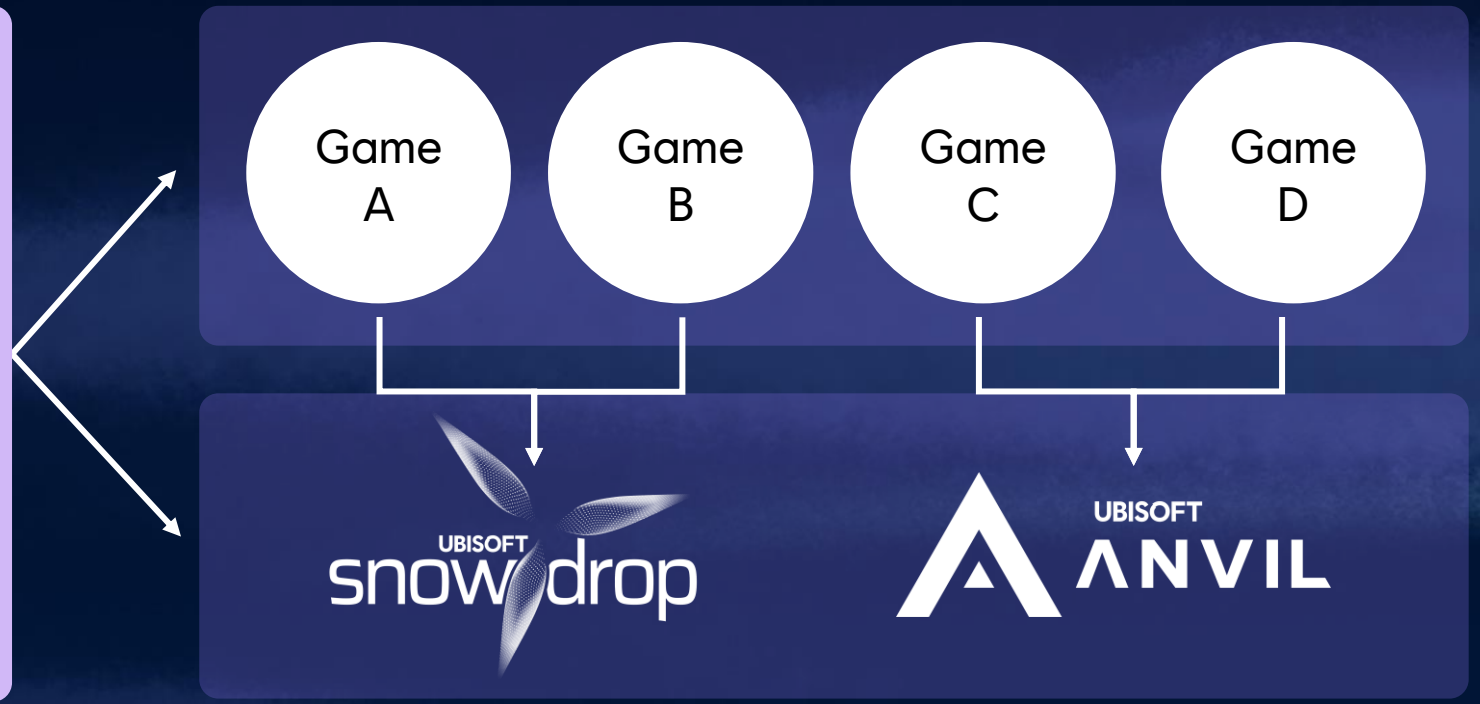
Recently: Games

Gameplay and AI: with machine learning.

AT UBISOFT...

Content Tools GROUP

Machine Learning Framework (MLF)



TAMING A BEAST

USING ONNX RUNTIME IN AAA GAMES

JEAN-SIMON LAPOINTE
TECH. LEAD AT MACHINE LEARNING FRAMEWORK (UBISOFT)

21 JULY 2025

WHY ML IN GAMES?

Input

- Position
- Goal
- Obstacles

ML Model

Output

- Move direction

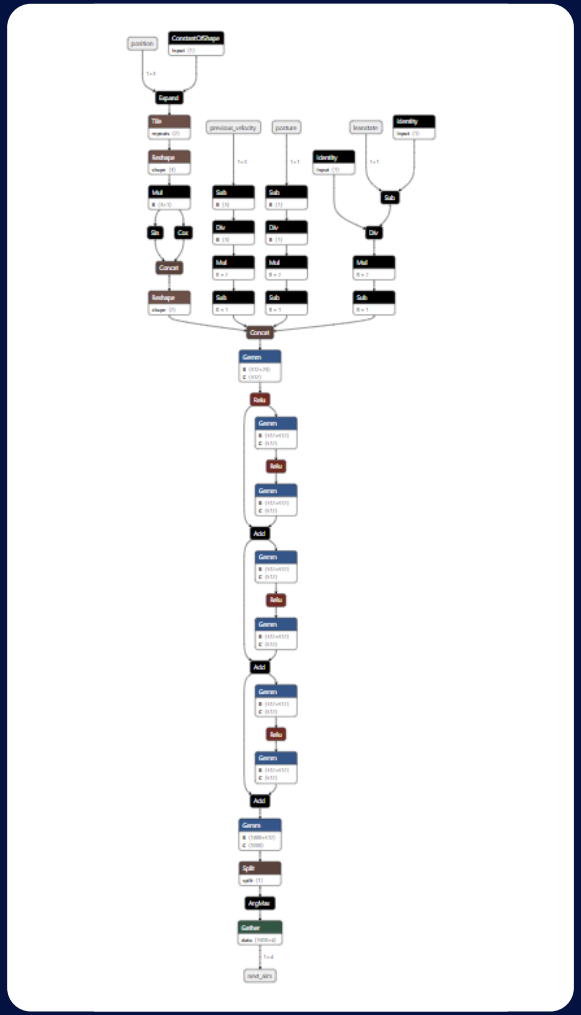
APPLICATIONS:

- Navigation
- NPC Fighting
- Speech Recognition
- Animation data
- Many others...

ADVANTAGES:

- Performance
- Better output results
- More natural/pleasant results

MODELS



Inference: Apply Model on inputs to get output

Data Scientists -> Trainings -> Models

- Format: Pytorch or TensorFlow(TF)

Onnx: Model File format

- Library to handle files.
- Open Source: Develop by Meta and Microsoft,
- Owned by Onnx community
- Easy conversion from Pytorch or TF

Onnx Runtime (OnnxRT): Inference engine

- Open Source: Developed and owned by Microsoft

ORIGIN

BACK IN 2019

Onnx Format,
starting to get traction

OnnxRT:
Still pre 1.0

Other inference engine
(Pytorch and TensorFlow)

Not well fitted for in game inference

RuNN

WE CREATED OUR OWN INFERENCE ENGINE

RUNN (RUNTIME NEURAL NETWORK) WAS BORN

Custom File Format

Basic set of operators for our current needs

Well Optimized

We were happy!

IS IT STILL ENOUGH?

2019

ML in games was still a new thing

- We controlled/knew all the users
- Users knew what we supported.
- We added operators when needed

2023

With scaling up of ML in games

- Models appear from everywhere!
- With operators that we do not support.

The inference engine was a bottleneck!

PIPELINE

ONNX Operator ‘Gather’ not supported yet.

- Onnx: > 180 operators
- Runn: ~ 40 operators

IMPLEMENTING OPERATORS IN RUNN

CAN BECOME A COMPLEX OF A JOB

- Subtleties with dimensions
- Can have many attributes that will affect the behavior
- A lot of use cases to test
- Some platform specific optimization

Typical time to implement one operator:

- Working version: between 1 and 30 Days
- Optimized version: up to 10 Days for each platform!

In the meantime: Data Scientist is Blocked

IN THE MEANTIME…

Onnx Format grew in maturity and popularity

Industry Standard

Onnx Runtime got more traction

In 2023:

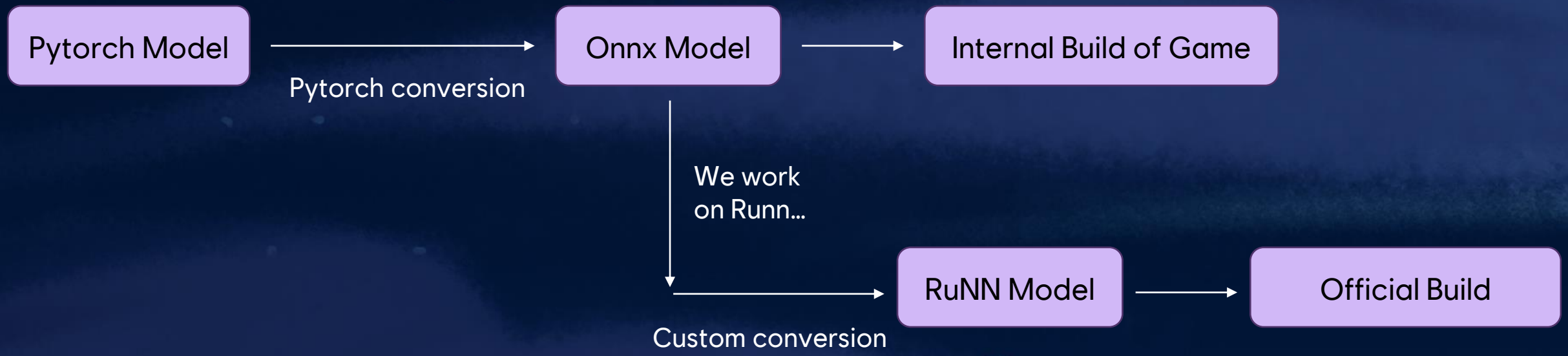
"Why can't we 'just use' Onnx Runtime in the game?"

- Anonymous Dev

It seemed like "Taming a beast"!

SOLUTION

ONNX RUNTIME AS NON-SHIPPABLE PACKAGE



SOLUTION

ONNX RUNTIME AS NON-SHIPPABLE PACKAGE

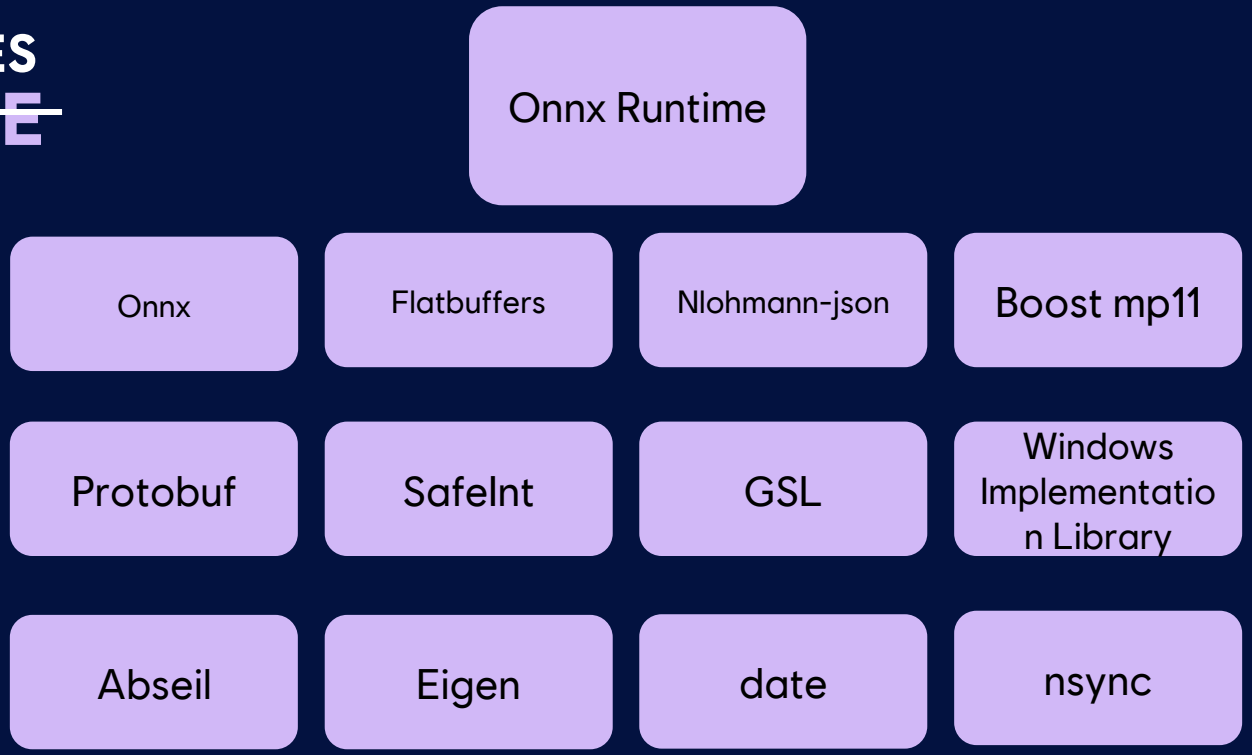
Windows build only: No console support

Build the library once and use it in the (non-shipped) games

LEGAL INTERMISSION I

IS IT OK TO USE ~~ONNX RUNTIME~~ ALL THESE LIBRARIES INTERNALLY IN NON-SHIPPED GAMES

- A lot of different licenses
  Apache 2.0, BSD, MIT and MPL2
- Need to ask your friendly Legal Dept.

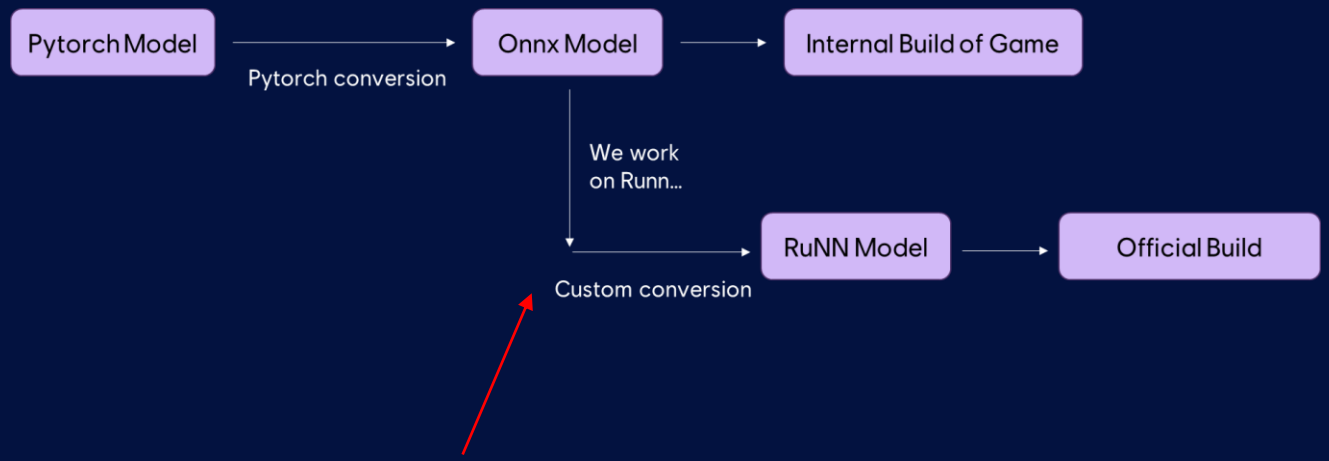


A WIN

A win!... Right?

It did help Data Scientists to be unblock.

- Many operators to implement and optimize
- It is still hard to keep up with a small team.



We were not ready!

WE NEED TO SHIP ONNXRT

What does it mean to actually ship a game with OnnxRT?

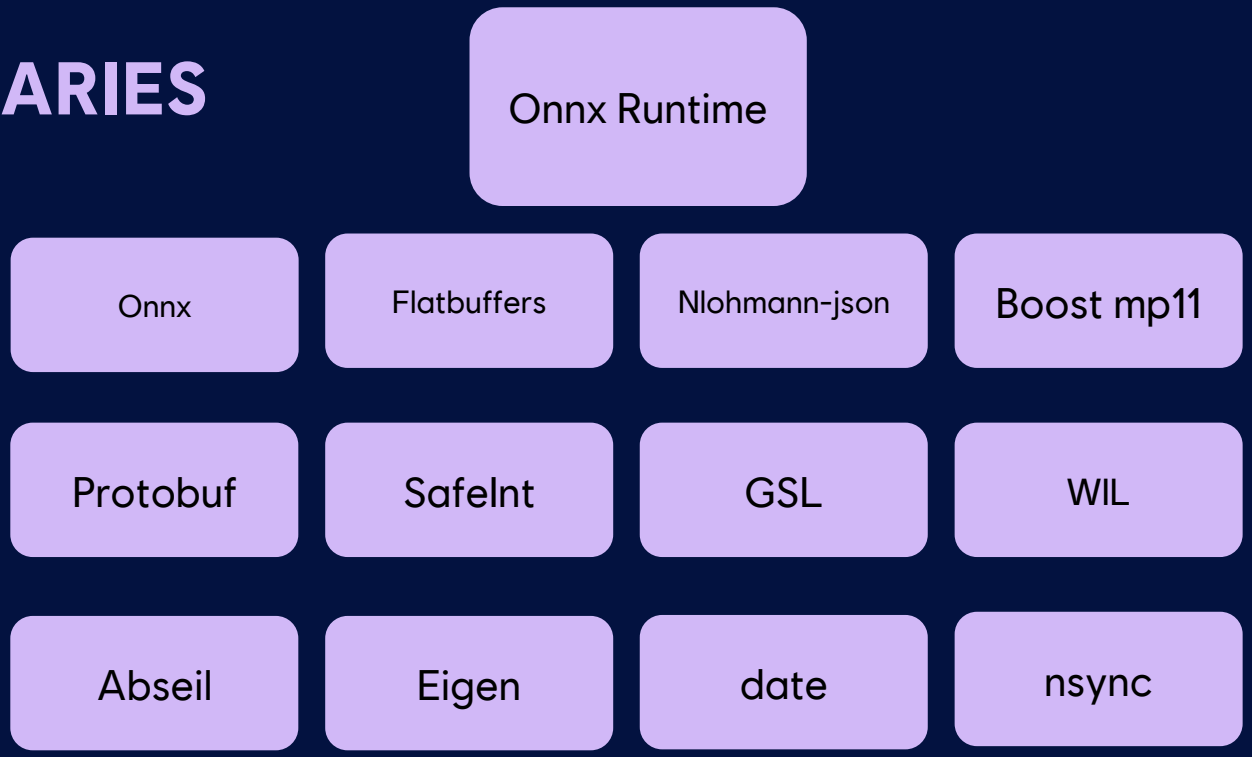
OnnxRT's code should be compiled with the game code

- OnnxRT's code becomes part of the game code
- It needs to comply with:
  - Build system
  - Architecture requirements
  - Platform specificities

LEGAL INTERMISSION II

IS IT OK TO USE ALL THESE LIBRARIES IN A SHIPPED GAME

- Still, A lot of different licenses
- Need to ask your friendly Legal Dept.



MEMORY MANAGEMENT

- Memory is a scarce resource
- ...Even more in games!
- Custom Memory Allocators
  - Keep track of the memory budget by package
  - Control on memory allocation: chunks, pool, arena...
- OnnxRT: Some support for allocator
  - Bigger Alloc., Tensors, ...
- All other allocations std:: and new
- Same for dependencies
- We can, and do, overload new!
- Lost of granularity
  - Memory used by ORT?
  - Transient allocations by ORT?
  - No Custom allocations for ORT

We needed to ask our engine architects

ARCHI APPROVAL

Code become part of the engine/game

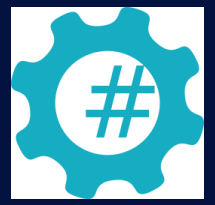
We need approval from engine architects first

- Memory management
- Warning level
- Just having that much new code in the build

Two engines involved in our AAA games

BUILD SYSTEM

- CMake is becoming de facto build generator script.
- Except in the gaming world…
- At Ubi, we have our own: Sharpmake
  - C#
  - OpenSource
  - CppCon 2017 Lightning Talk by Nic Fleury.



- One need to convert all the CMake scripts into your favorite build system.

BLOB BUILD

BLOB/UNITY/JUMBO BUILD

- One jumbo cpp file to rule them all.
- Saves Build times.
- Efficient for templated classes and functions
  - Compile std::vector<int> 20 times!
  - Same for all std library
  - And for your own low-level libraries
- Blob build speed-up: 2 to 5 times faster.
  - Even compared to PCH

```
#include <A.cpp>

#include <B.cpp>

#include <C.cpp>
```

BLOB BUILD

Unnamed namespace

```cpp
// FileA.cpp

namespace FileA_Private {
std::string GetElementName() {
  // Compute FileA element name
  std::string name = "";
  return name;
}
} // namespace FileA_Private

// …
auto myName = FileA_Private::GetElementName();
```

```cpp
// FileA.cpp

namespace {
std::string GetElementName() {
  // Compute FileA element name
  std::string name = "";
  return name;
}
} // namespace

// …

// FileB.cpp

namespace {
std::string GetElementName() {
  // Compute FileB element name
  std::string name = "";
  return name;
}
} // namespace

// …
```

EXCEPTION

Handy C++ construct to help deal with error handling

Considered Evil in the gaming world!

Performance Issues

Lucky: We have a Disable Exception flag

It replaces throwing an exception with… Abort! -> Crash

EXCEPTION

Most abort calls are preventable

- Bad usage: Preventable with more testing

Other cases:

- Error handling needs to be done outside.
- We might need to add some functions

```
GPUResource * GetResourceIfAvailable();
```

```
GPUResource * DeepResourceGet() {
  ORT_try {
    auto resPtr = LowerLevelGetResource();
  }
  ORT_catch (…){
    return nullptr;
  }
}
```

```
bool CheckResourceIsAvailable();
```

THREADING

In Games, we prefer to manage our own threads.

- Limit number
- Full control (Affinity and priority)
- Better Monitoring

OnnxRT creates threads for Parallel execution.

- Provides flag to enable/disable thread creation

But we lose the potential benefits from the parallel execution of inference.

Parallel optimization is not that beneficial, when all cores run at full capacity.

PLATFORM SPECIFIC

BUILDING FOR PLATFORMS

Consoles, Windows, Linux, Mac, Ios…

Some live games still need to support old setup

Old Consoles, Old Version of Windows…

Some consoles do not have full support of standard C and C++ features.

EXAMPLES:

- Stubbing: File Creation
- API Adaptation: localtime_s().

PLATFORM SPECIFIC

```
static PerThreadObject* GetPerThreadObject() {
   static thread_local PerThread per_thread;
   PerThreadObject * pt = &per_thread;
   if (!pt->initialized) {
      pt->Init();
      pt->initialized = true;
    }
   return pt;
}
```

```
EigenNonBlockingThreadPool.h(1696,35): error : alignment (128) of thread-local variable 'per_thread_' is greater than
the maximum supported alignment (32) for a thread-local variable on this target
```

Simple issues but Numerous.

```
static PerThreadObject* GetPerThreadObject() {
   static thread_local std::unique_ptr< PerThreadObject > per_thread;
   if (!per_thread) {
      per_thread = std::make_unique< PerThreadObject >();
      per_thread->Init();
      per_thread->initialized = true;
   }
   return per_thread.get();
}
```

DEBUGGING

Widely used software
=> Less bugs

Yes, but… When there is a bug!

Much harder to investigate.

The code is huge

No in-house expert

Ex.: Random Crash

Multiple days of debugging

Allocator alignment

Debugging becomes much more involve

Increase Tech. Debt

PERFORMANCE

RUNN compatible models

- Big Models: ONNXRT >> RUNN
- Smaller Models: RUNN > ONNXRT

Other models

- Good performance navigation models
  - Frequent inferences, small networks, navigation
- Tests with bigger models.
  - Runtime budget not met
  - Bad perf with concurrent inferences.

Article from EA about OnnxRT:

- Would be "Not suited for AAA games"

> Generally, inference support in game engines is relatively new and improving rapidly with respect to performance and compatibility with other game systems. Integration with other systems frequently requires operating on a batch size of 1 which means that batching for speed-ups in inference is not always possible. Inference run-time is critical for production, where current solutions for automated testing only require $<100\,\mu s$ for decision making, setting an upper feasibility boundary for inference time. The ONNX runtime provides a good starting point, but for AAA games that would like to use ML models for many features, or for many agents, a bespoke high-performance inference library will probably be needed. In our experiments, we were limited to fairly small models due to previously mentioned restrictions ranging from 10^3 to 10^6 parameters in size. As inference support improves, these models could be increased in complexity. Furthermore, quantization of model parameters is a good way of increasing performance and reducing memory footprint but it was not used in our experiments.

PERFORMANCE

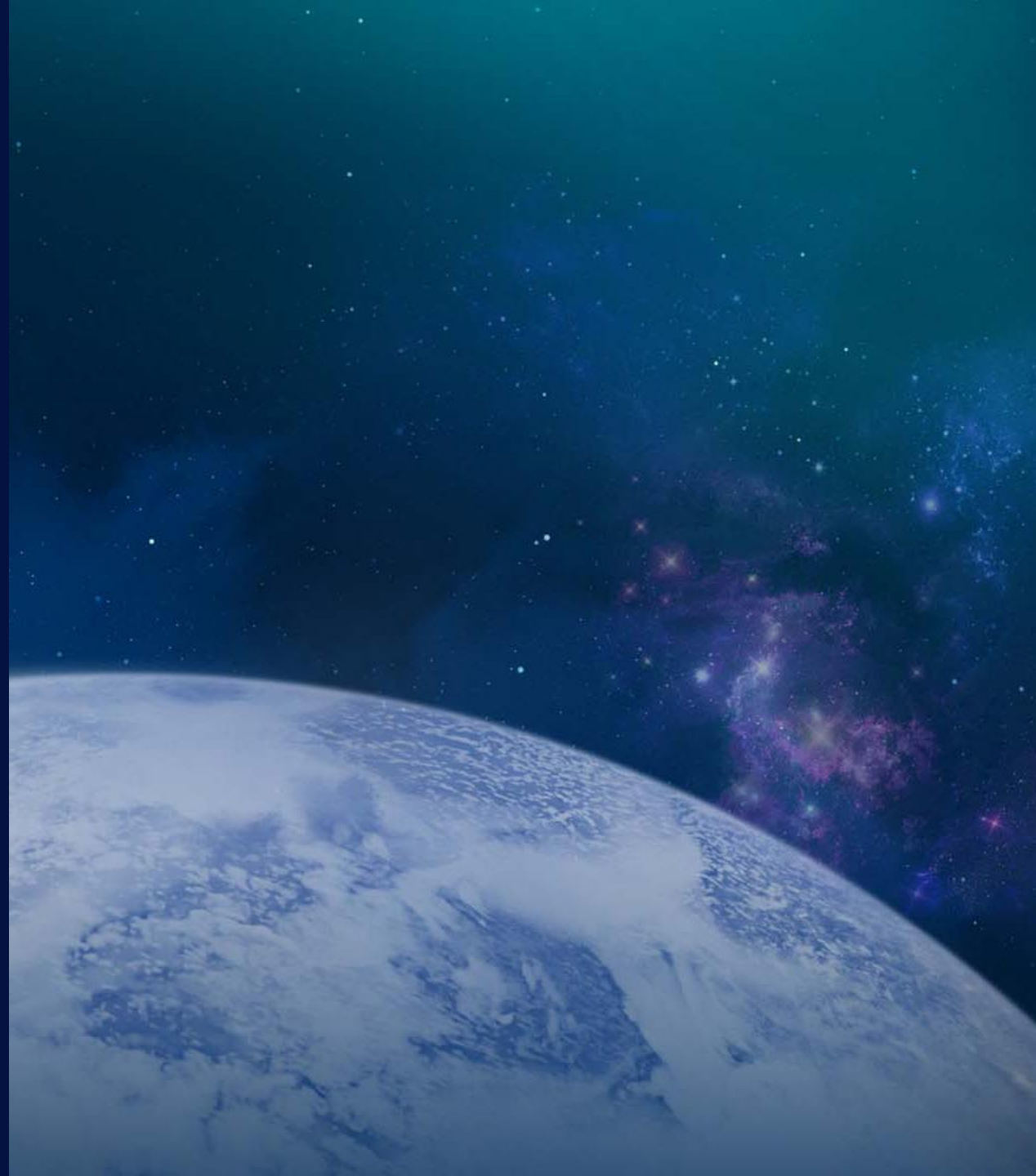
Non-trivial to get a proper broad performance analysis

Nothing to compare to

Bad performance could be Independent from the library

- Could be due to Model design
- Choice of operators
- Size of Operators
- Quantization

Ongoing investigation of performance



BUT WE MADE IT

Shipped in a game.

Integrated in all engines.

Ready to ship in new use-cases, in new games.

WHAT IF ONNXRT BUMPS ITS VERSION?

CONFLICTS ON THE HORIZON

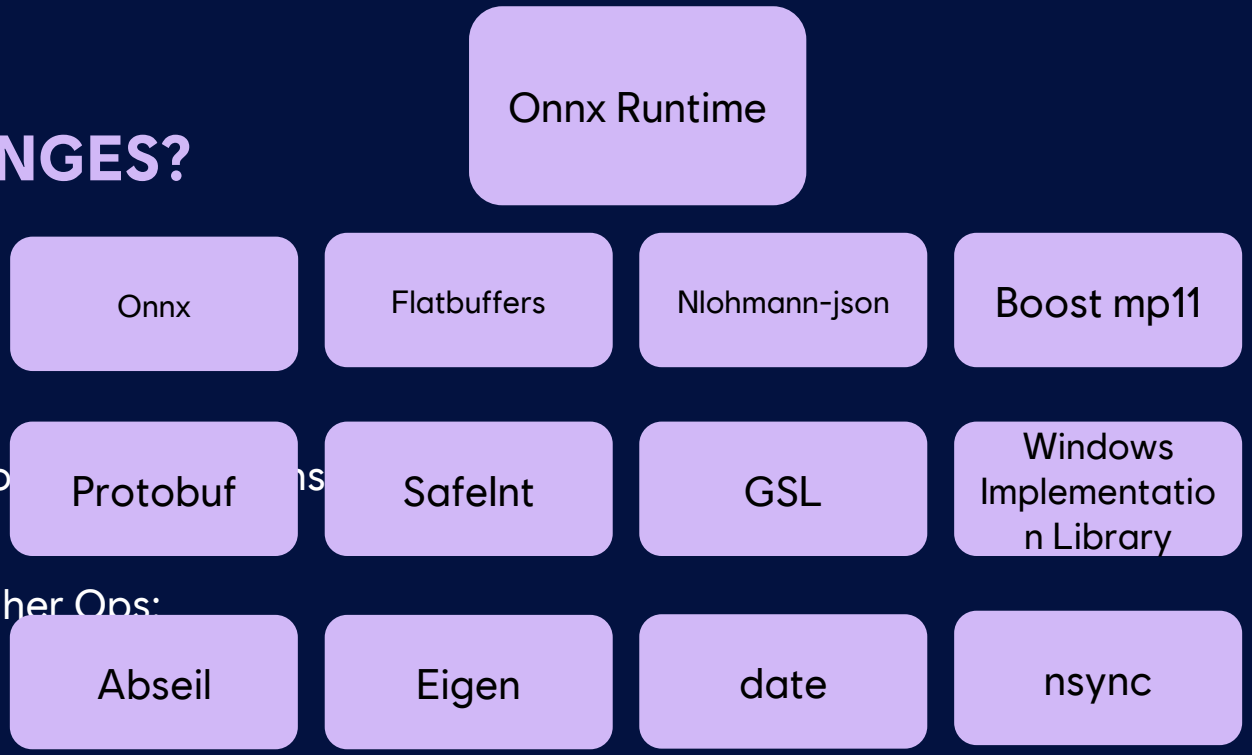
WHAT IF ONNXRT BUMPS ITS VERSION?

- Build system changes are isolated: Easy
- Changes are disseminated in multiple files
- Changes in multiple repositories (in the dependencies)
- Upstreaming our changes?
- The changes could in fact benefit other users
- Would minimize conflicts when we update.

LEGAL INTERMISSION III

IS IT OK TO UPSTREAM OUR CHANGES?

- Only some repositories needs changes:
- As of today, we got changes in 3 repos.
- Each repo has its own contribution guideline and Co ns Agreement (CLA)
- CLA will typically need approval/signature from higher Ops:
- Expect Some delays …



LEGAL INTERMISSION III

```
unsigned int GetThreadId() {
#if defined (__APPLE__)
   // Do Apple stuff
#elif defined (_WIN32)
   // Do Windows stuff
#endif
}
```

```
unsigned int GetThreadId() {
#if defined (__APPLE__)
   // Do Apple stuff
#elif defined (_WIN32)
   // Do Windows stuff
#elif defined(_CON
   return static_c
#endif
}
```

```
unsigned int GetThreadId() {
   ///…
#elif defined(ORT_USE_PTHREAD) && defined(ORT_THREAD_ID_IS_ULONG)
   return static_cast<unsigned int>(pthread_self());
#endif
}
```

DO NOT LEAK YOUR PARTNERS' STUFF

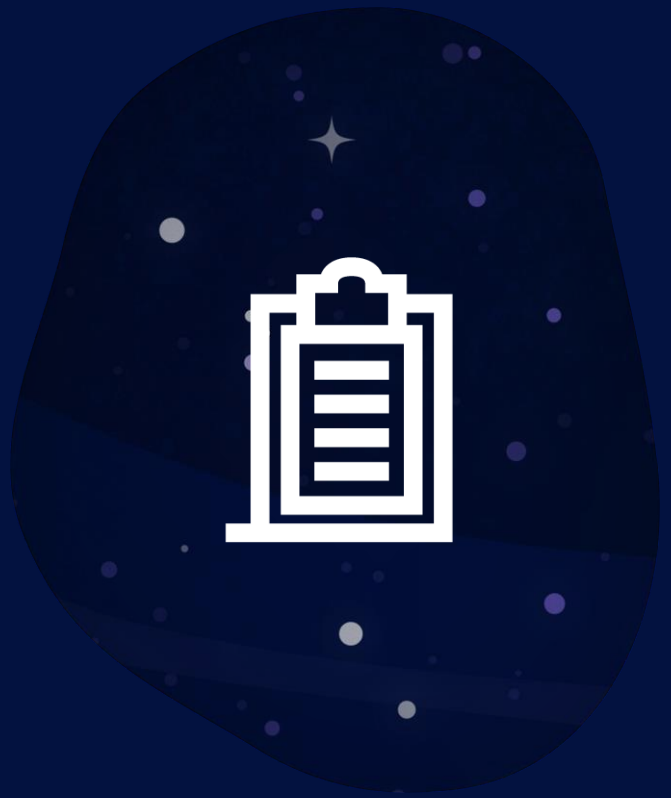
- Partners = Hardware manufacturer and SDK providers
- The agreements with partners are solid and crucial
- We do not want a glimpse of possibility to break them

Better safe than loosing partnership!

ACTUALLY MERGING

WHEN TIME COMES TO UPSTREAM

- Make sure changes are digestible
  "One topic" merge request
- Convince code-owner that changes are valuable
- Convince code-owner that changes are worthy of their time
  Merge request Limbo
- Changes are sometimes not merged in a reasonable amount of time…
- This will affect your next upgrades!



WHAT ABOUT THE BEAST?

Have we tamed the beast?

Yes!

- Rainbow 6 Siege has shipped with player facing navigation using OnnxRT
- We have other games actually using OnnxRT not out the door yet!
- More and more users trying ML in games

But

- Some set back with a game: not respecting budget
- Tech debt: Debugging and upstreaming

"Don't reinvent the wheel"?

Almost always true.

Think about all the factors:

- Performance
- complexity


THANK YOU!