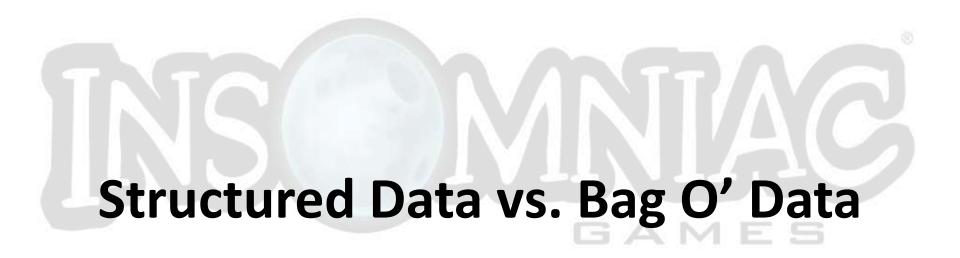


Just because you put your data in a struct doesn't mean your data is structured



Common pitfalls and easy tips.

Don't mix context information with data to be transformed.

These are two *unrelated* sets of data.

```
class InternalGridAttribute : public BaseInternalAttribute
31 📥
32
33
     public:
34
35
36
                 m Visible;
37
        bool
38
39
                 m Enabled;
        bool
40
       // API
43
44
       inline uint32_t GetGridSizeSetting( void ) const;
45
       uint32 t
                        uint32_t Luna::InternalGridAttribute::GetGridSizeSetting(void) const
46
                                                              GAMES
```

Example 1: How does the above relate to the below?

```
// Internal Data
protected:

uint32_t m_GridSizeSetting;
Grid m_LocalGrid; //!< Draw the major and minor line grids

static const InternalAttributeFieldInfo m_AttributeFields[];

static const InternalAttributeFieldInfo m_AttributeFields[];

};
```



Example 2: Mixing state machine state and transformed data

```
40
      public:
        enum modifiedState
43
44
          kUnmodified,
45
          kModified.
46
          kModified NeedResolve,
47
          kModified NeedDDLUpdate,
48
49
50
        Handle
                                  m Handle;
51
52
        DDLInstance::MemoryCB*
                                  m InstanceMemoryCB;
                                                         // Pointer to the DDL instanc
53
        DDLInstance::Instance*
                                  m Instance;
                                                         // Pointer to the DDL instanc
54
        AttributeDef::Index
                                  m DefIndex;
55
56
        AttributeDB*
                                  m DB;
57
58
        inline bool IsInternalAttribute( void );
59
60
        inline ModifiedState GetModifiedState( void );
61
                                                         Example 3: Public, protected, private data
62 📥
        // Internal Stuff
63
64
      protected:
65
       ModifiedState
                                m ModifiedState;
66
67
68
       BaseInternalAttribute* m InternalData;
69
        friend class BaseAttributeAccess:
70
71
72
        friend class AttributeDB:
       inline Attribute* FreeListGetNext( void ) const;
73
74
        inline void FreeListSetNext( Attribute* next );
```

There's no such thing as "optional" data.

"Optional" is context information.

Separate that data.

Don't mix data with different lifespans.

"Lifespa You'll spend checking to se

Example:
Update classes.
Build-time data versus
runtime data.

For each vs. For all

"For each" considered harmful.

What if branching was one of the most expensive things you could do on the CPU? ...oh, wait.



SOA by default.

Combine as *appropriate*.

Easy Step #1: SOA

Combine based on use: e.g. xyzw

iate.

Easy Step #1: SOA

Con

What about conflicting combinations?

This problem *already* exists.

Easy Step #1: SOA

Con Wł

Can solve for cache coherency.

~2B PPU cycles/sec wasted on waiting for memory.

This r

And, of course... Concurrency.

Separating data by context makes it much simpler to solve for data concurrency.

