Creating Your Building Blocks
Modular Component AI Systems

- Brett Laming, Rockstar Leeds
- Joel McGinnis, CCP
- Alex Champandard, AiGameDev.com

AI
ARTIFICIAL INTELLIGENCE SUMMIT

Overview

1. Brett Laming
   - Component systems revisited
2. Joel McGinnis
   - Behaviour and Design Patterns
3. Alex Champandard
   - Performance and Multi-threading

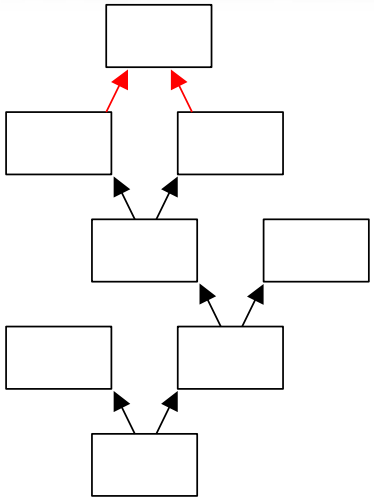
Part 1. Brett Laming

COMPONENT SYSTEMS REVISITED

Component Systems

- What are they?
  - No single definition
- Potentially
  - Smart objects
  - COM
  - Game object / entity architectures
  - Plug-ins
  - Message based, data driven
- Fairly certain `class cOgre : class cMonster` is wrong

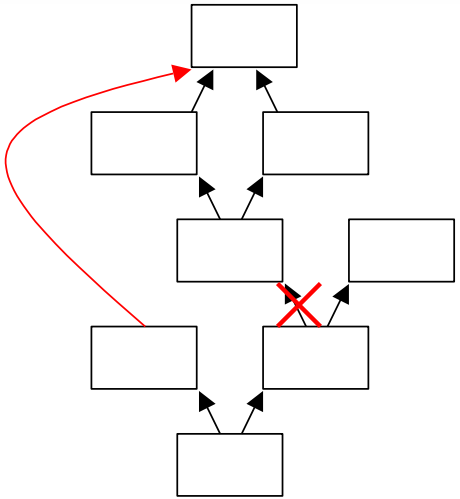
Background



DEEP CLASS

```
class cThrowingKnife :
public cRangedWeapon,
public cMeleeWeapon
```

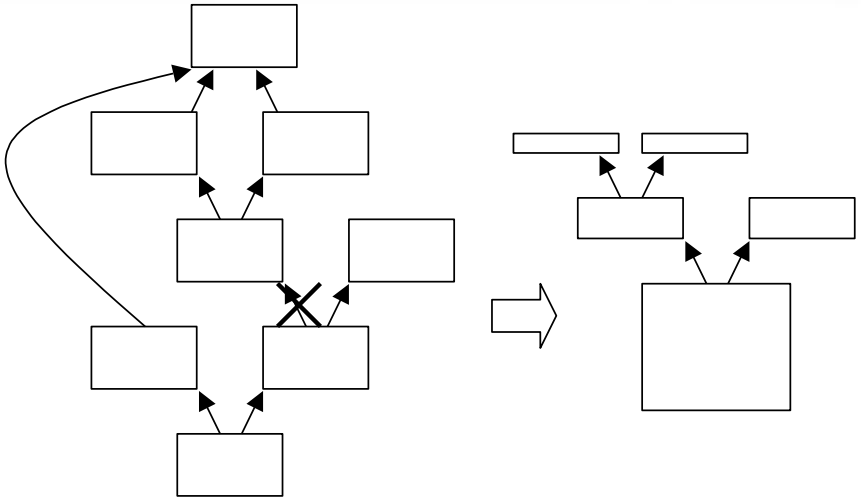
Background



DEEP CLASS

```
class cWeapon : public cDynamicProp
class cRangedWeapon : public cWeapon
class cBow : public cRangedWeapon

class cBallista :
public cRangedWeapon,
public cStaticProp,
!public cDynamicProp
```

Background

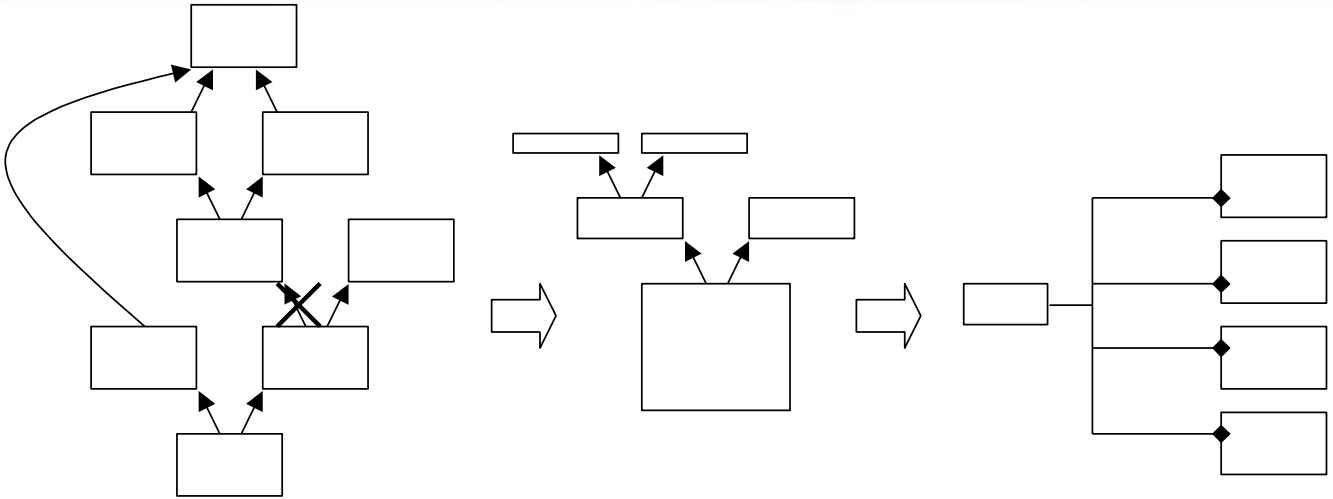


DEEP CLASS

FAT CLASS

```
class cWeapon : public cGameObj
{
     cGameObj*     CreateAmmo();
     // Reloading not for melee
     eState        mState;
     eAttackMode   mAttackMode;
     eAmmoType     mAmmoType;
     // Ranged weapons only
     int           mAmmoCount;
};
```

Background

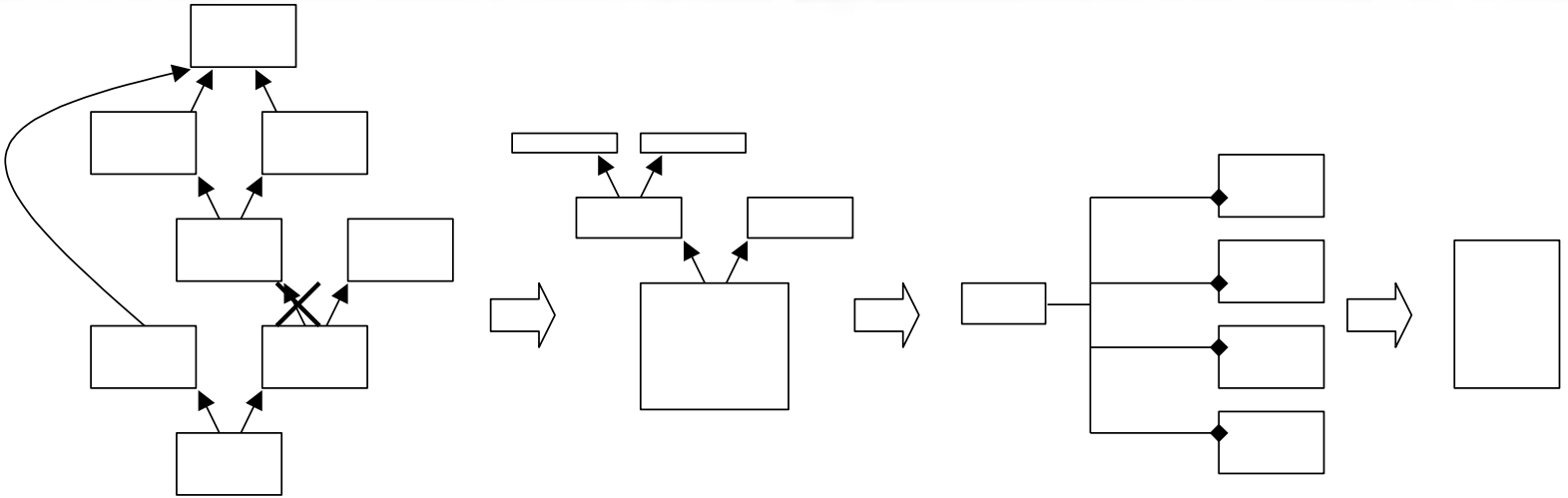


DEEP CLASS

FAT CLASS

PLUGIN

Background



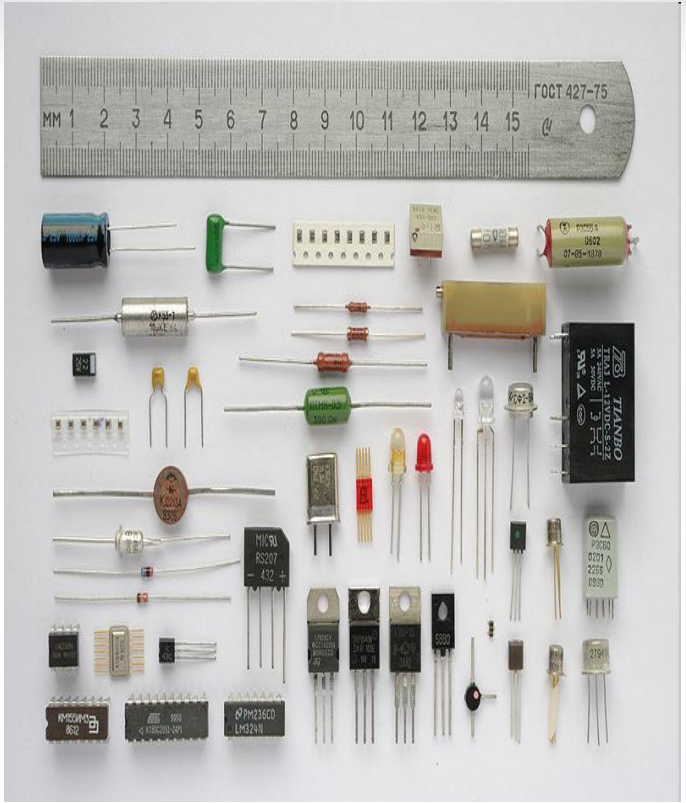
DEEP CLASS FAT CLASS PLUGIN DATA DRIVEN

Damned if you do…

- Don’t believe it.
- We *get* the problems

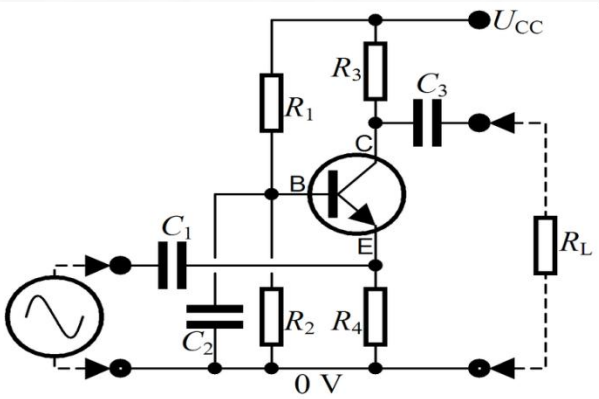
Component

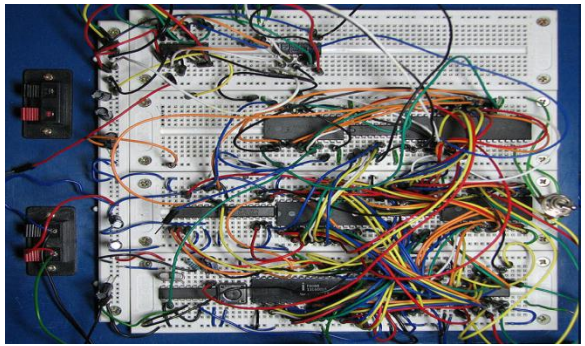
- Broad Classification
- Key Properties
- Defined I/O
- Interchangeable

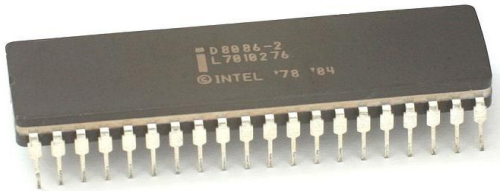


System

- Organisation
- Compartmentalization







Reusable A.I.

- Output → gameplay.
- Input ← gameplay world
- ∴ Disciplined gameplay
  - Good organisation
  - Purposeful data
  - Sensible lifetimes
- ⇒ Good reusable A.I.

5 key levels of organisation

INHERITANCE
- Taxonomy
- Component Name

STRUCTURE
- World Organisation
- Parents – Children

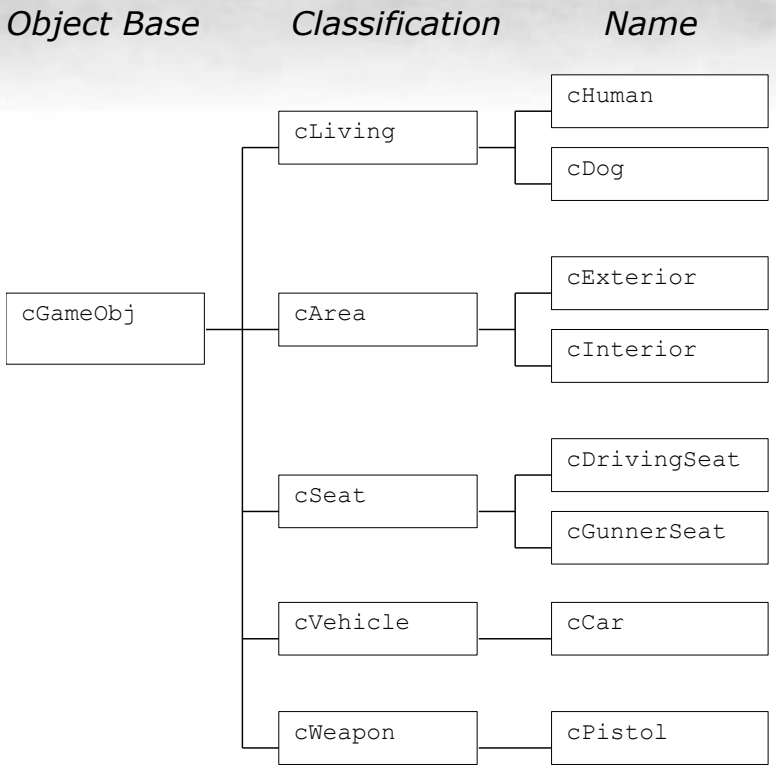
DATA FLOW
- A.I. → Gameplay

PARALLELIZATION
- Homogenous
- Batches / Jobs

COMPARTMENTALIZATION
- Data boundaries
- Smart objects and DLC

Inheritance

- Classification
  - RTTI queries
  - Ability to sort by class
- Name
  - RTTI factory creation
  - Ability to serialise
- Combined
  - Data driven approach
  - Shallow hierarchy



RTTI Power

```
typedef int RttiType
DECLARE_RTTI_TYPE
IMPLEMENT_RTTI_META_BEGIN
IMPLEMENT_RTTI_META_END
RTTI_CLASSIFY_AND_ADD( mpSeat, cSeat, p_obj );
cWeapon *p_wep = DynamicCast<cWeapon*>(p_obj);
cRegistry::Instance().Create( R_STR("cColt45") );
virtual void Serialise( cAttributeReader &rdr );
virtual void Serialise( cAttributeWriter &wtr );
rdr << PTR_IS_OWNED( mpSeats )
```

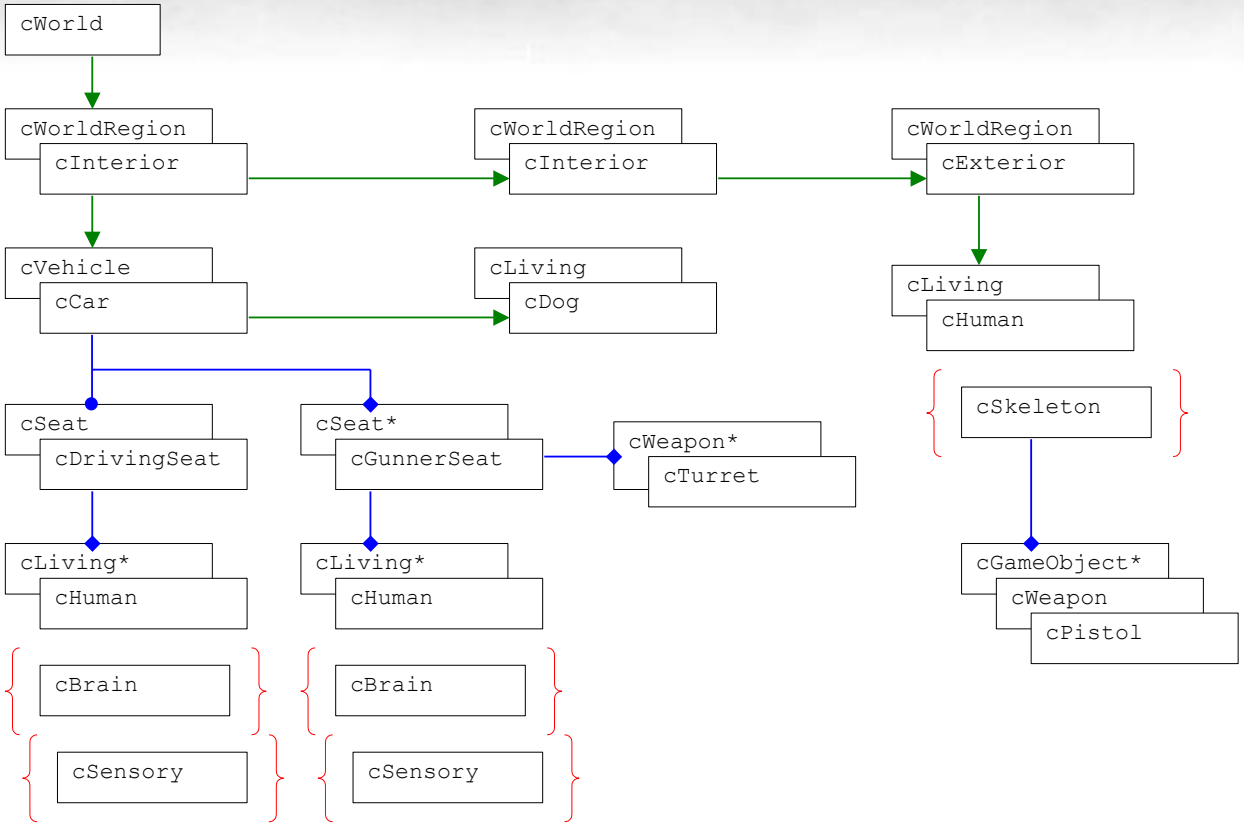
- With a pre compile step, you can make it extremely efficient indeed!

Structure

- Spatial
  - cGameObj
  - Reference frame
  - World transform
- Functional
  - Composition
  - Aggregation
- Dependency tracking
  - Conflict resolution
  - Job ordering

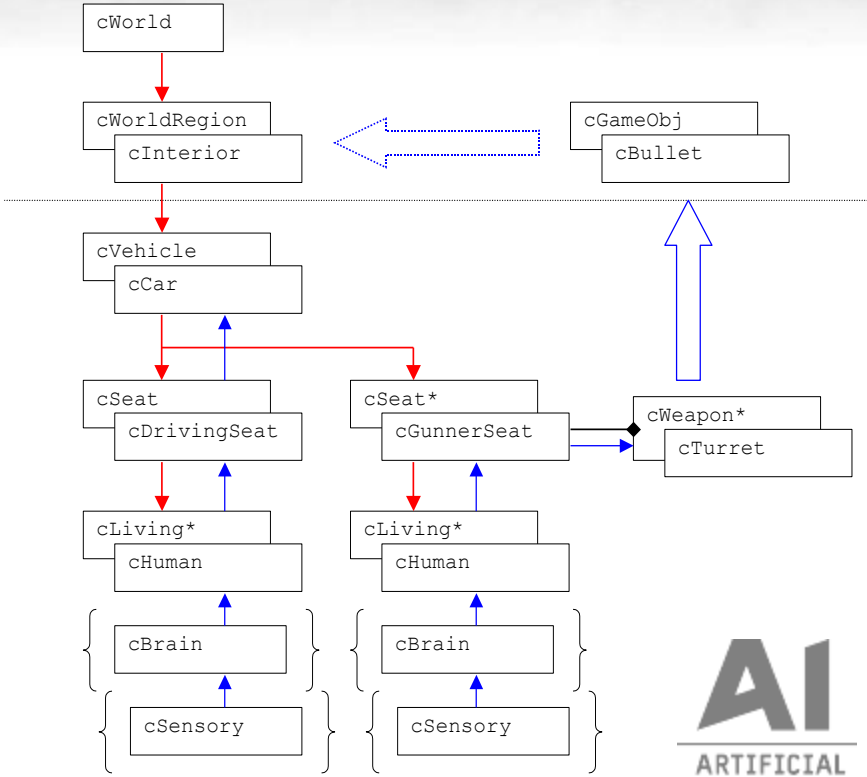
```
class cThing
{
      RttiType mRTTI;
};

class cGameObj : public cThing
{
public:
private:
      cGameObj          *mpParent;
      cGameObj          *mpFirstChild;
      cGameObj          *mpNextSibling;

      cMat4              mLocalTransform;
};
```

Structure & Inheritance



Data Flow

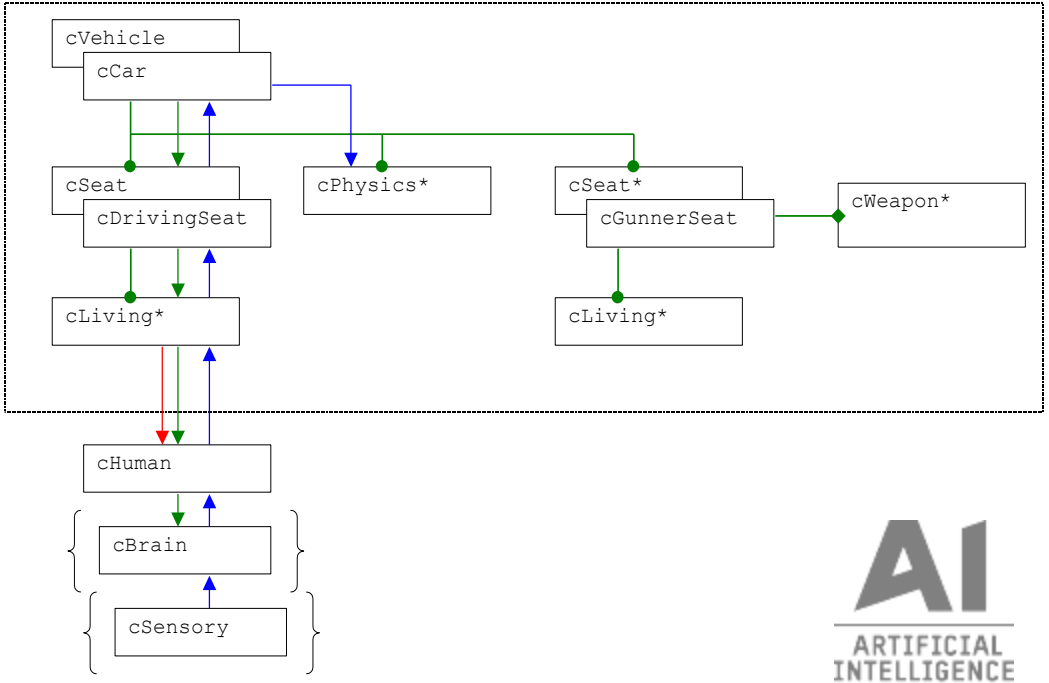
- Data Flow
  - World State → A.I → Gameplay → World State
- Changes to structure
  - Not inside dt!
  - Upstream ⇒ Message
  - Downstream ⇒ Message
- Changes to properties
  - Downstream ⇒ Signalling
  - Upstream ⇒ Signalling
  - Spatial Barrier ⇒ Message



AI ARTIFICIAL INTELLIGENCE SUMMIT

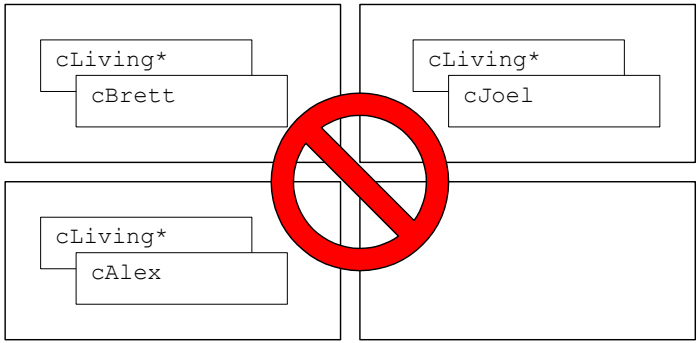
Compartmentalization

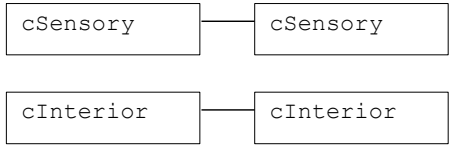
- Smart Objects
  - Reconstructable by RTTI
- Near free
  - Given good structure
- External instructions
  - A.I., animation etc…
  - Carried by signalling



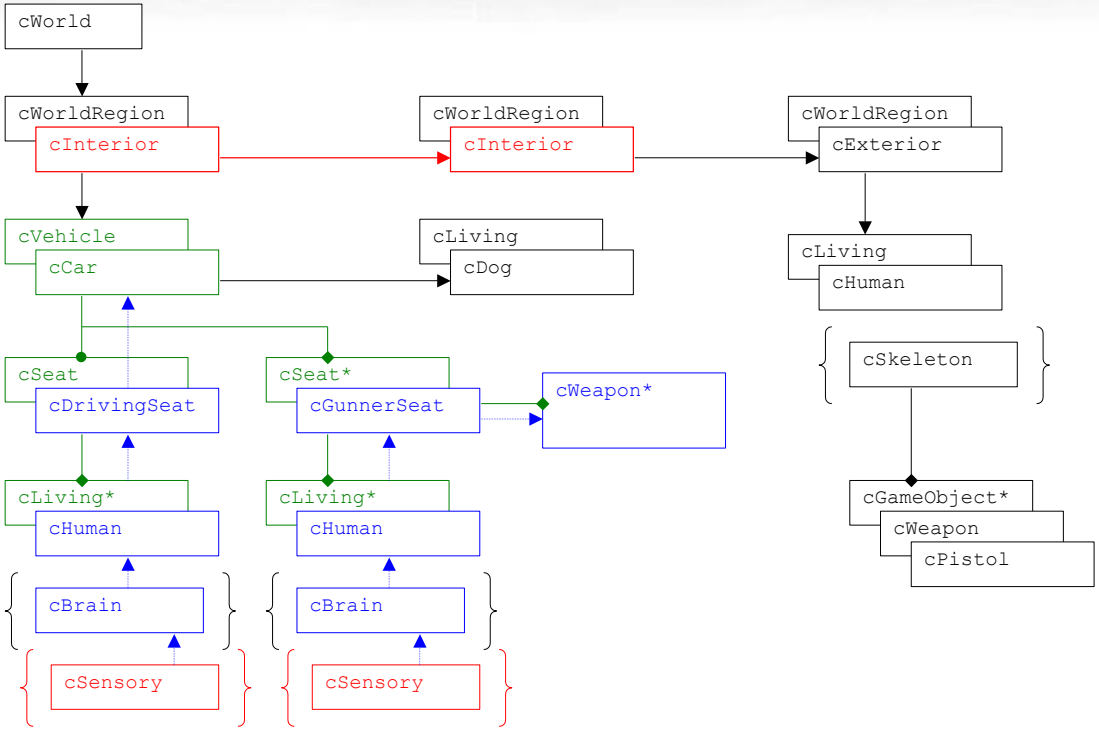
Parallelization

- The ideal…
  - … is still a way off
- A.I./gameplay still parallelizes!
  - Even in game graphs!
  - Indirection
  - Aliasing
- Candidates
  - Leaf output
    - animation, navigation, component update
  - Leaf input
    - sensory info, blackboards, ray tests





All things being good…



ARTIFICIAL INTELLIGENCE SUMMIT

Design Tricks 1

- Remove temptation
  - Minimal data
    - Per frame ⇒ stack
  - Minimal lifetime
    - Use new/delete boundary!
    - Pools
- Favour derivation
  - No equation contradiction
  - No duplicate data
- Potential deep class problem?
  - Generalise

```
class cPhysicalProperties
{
public:
      inline float Volume() const;
      inline float Mass() const
      {
            return Volume() * mDensity;
      }
      inline float BoundsRadius() const;

      inline bool IsCarriable( cAABB grasp,
                               float force ) const;

      inline bool IsThrowable( float force ) const;

private:
      cAABB mBoundingBox;
      float mDensity;
};
```

Design Tricks 2

- Locality of reference
  - Abstraction + composition
  - Placement new
  - Embedded lists
  - Pools
- Minimise NULL checks
- Non-virtual pathways
  - Use RTTI filtering
- Many virtual pointers
  - Package once and carry downstream

```
class cProjectile : public cGameObj
{
public:
      DECLARE_POOL( ... );
      cProjectile() : mpPhysics( &mNullPhysics )  { }

      void SetGravity( ... ) { mpPhysics->Add( mGravity ); }

private:
      iPhysics *mpPhysics;
      cGravity mGravity;
      static cDummyPhysics mNullPhysics;
};
```

Conclusions

- Gameplay gives us fun buttons to press!
  - Tight game-play ⇒ Good, reusable A.I.
- Think
  - Minimal classes
  - Data life time
  - Locality of reference
- Use
  - Generalisation
  - RTTI
  - Placement new/delete
  - Pools
- Nothing is really that un-surmountable!

Part 2. Joel McGinnis

AI DESIGN PATTERNS

What are the pressures?

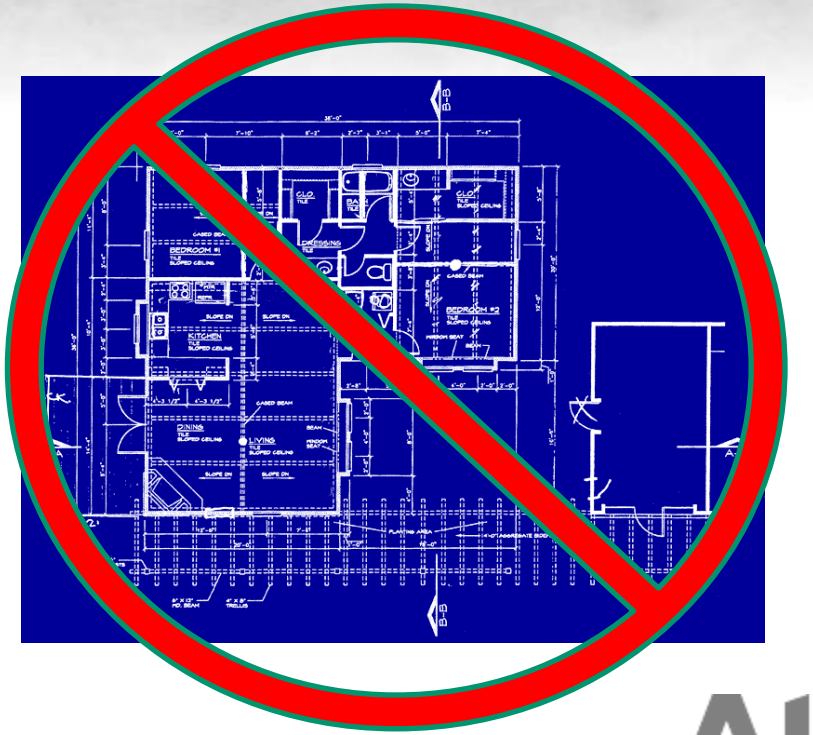
- Resources
  - Cycles
  - Memory
- Design specificity

CA for AI

- Flexibility
- Performance balancing

Word of warning

- Paradigm not architecture



- So we'll be looking at patterns

ARTIFICIAL INTELLIGENCE SUMMIT

TAKING IT APART

(anti)Pattern

“Where shall we put the data?”

“Lets just put it on the AIComponent”

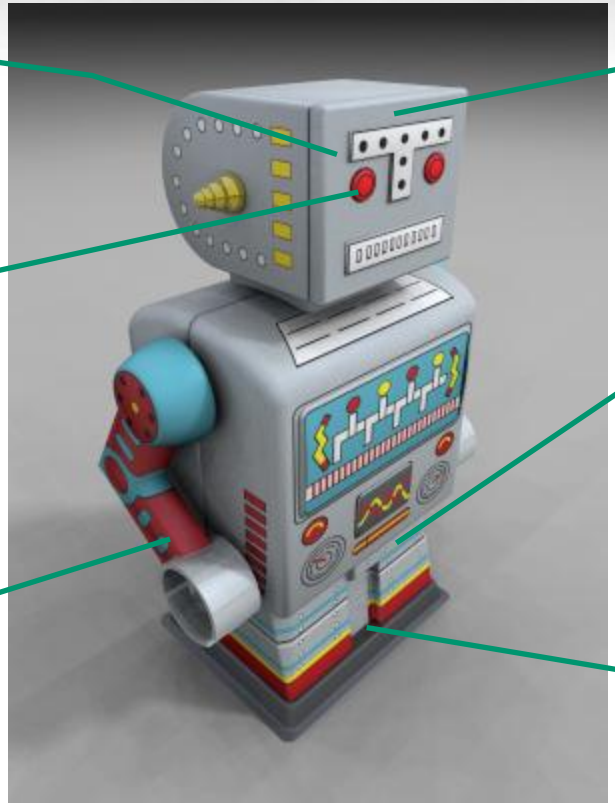
“That seems like a bad idea, lets not do it”

So what do you have?

Target manager

Perception

Tracking



Behavior Tree

Pathfinder

Movement controller

What you consume

- Focal point
- Targetable object
- Cover markup
- Interaction point
- Trigger volume

- Granularity is Good!

Component matrix

PUTTING IT BACK TOGETHER

Substitution

- Perception
- Behavior Tree
- Pathfinder
- Targeting
- Animation
- Standard movement

Substitution

- Perception
- Behavior Tree
- Pathfinder
- Targeting
- Animation
- Big creature movement

Substitution

- What did we gain?
  - Wasn't enough to ship but...
    - Minimal investment
    - Nice prototype
  - Answered design questions sooner
- Required:
  - COM, signaling, interface, messaging
- Leverage hierarchy
  - OOP under the CA

Find Via Registration

Target Selection

Targetable

Find Via Registration

Target Selection

Targetable

Targeting System

Find Via Registration

Target Selection

Targetable

Targeting System

Find Via Registration

- What did we gain?
  - Reduced search space
  - Scoping
- Simplify construction of behavior
- Required:
  - Life-cycle management

Late construction of types

Target Selection

Targetable

Late construction of types

Target Selection

Targetable

Late construction of types

- What did we gain?
  - The ability to change our minds
  - Load balancing
    - Try it everywhere
    - Keep it where most effective
- Required:
  - Data driven(?)
  - Light weight

Things to keep in mind

- Simplest affordances – greatest benefit
- Prefer small and light-weight CA
  - Lots of little components

Part 3. Alex Champandard

PERFORMANCE & MULTI-THREADING

You Must Be Wondering…

"How do you reconcile this modularity with high performance on all hardware?"

Demo Interlude

- Example Component
- Influence Maps
  - Come back at 3:00 for details!

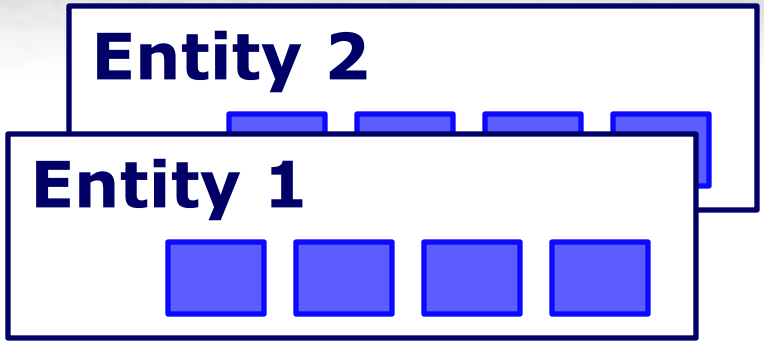
High-Performance

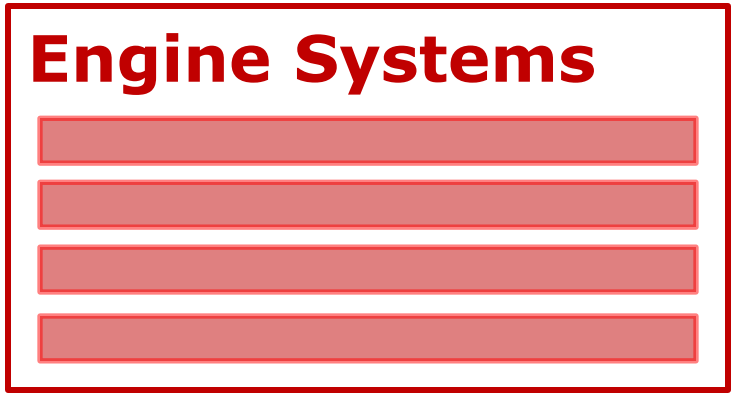
- Vectorization
  - Update 4x maps at a time with SIMD.
- Parallelization
  - Run batches of 4x maps on multiple cores.

The Solution

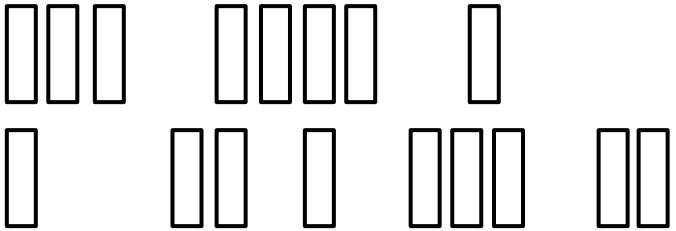
- Build your Engine as modularly as Entities!
  - Physics, Sensory, Reasoning, Behavior, Navigation, Locomotion, Animation.
- Just assure the break-down is the same.
  - It opens up opportunities for optimization.

Architecture





Jobs



Section 2.

BREAKDOWN

Component: Configuration

```
# Threat Type, Influence

HeavyEnemy      = 4.0
RangedEnemy     = 2.0
MeleeEnemy      = 4.0
ScoutEnemy      = 1.0
```

Component: Interface

```
class ReasoningComponent
{
public:
    void setEntityThreat(EntityId, float threat);
    void setAreaThreat(AreaId, float threat);

    float getAreaThreat(AreaId) const;

    /* ... */



};
```

Component: Communication

```
class ReasoningComponent
{
public:
    void setEntityThreat(EntityId, float threat);
    void setAreaThreat(AreaId, float threat);

    float getAreaThreat(AreaId) const;

    /* ... */

    typedef Delegate<void (float)> ThreatObserver;
    void notifyThreatLevel(float threshold, ThreatObserver);
};
```

Component: Life-Cycle

- Request new influence map on init().
  - Or when entering combat state.
- Remove it on shutdown()
  - Or when going into wounded state.

System: Batching & Prioritization

- Don’t process individual requests…
- Instead decides how to spawn jobs
  - Group maps updated at same frequencies.
  - Limit maximum number of jobs per frame.

System: Memory Allocation

- Manage memory for all influence maps.

- Customize allocation:
  - Allocate 4x maps at a time!
  - Interleave the float values for SIMD.

Jobs: Workload

- Implemented using SSE, Altivec.
  - Process 4x maps at a time
- Output
  - Influence Data
- Input
  - Level Map
  - Parameters

Jobs: Parallelism

- Jobs are isolated from each other.
  - No communication or inter-dependencies.
- Can run in parallel if necessary.

Section 3.

SUMMARY

AI
ARTIFICIAL
INTELLIGENCE
SUMMIT

Components

1. Very lightweight
2. Simple interface
3. Handles events
4. Data-driven

Systems

1. Memory Allocation
2. Computation Limits
3. Batching
4. Prioritization

Jobs

1. Computationally heavy work
2. Easily parallelizable code
3. Clear interface w/ engine

Creating Your Building Blocks
Modular Component AI Systems

- Brett Laming, Rockstar Leeds
- Joel McGinnis, CCP
- Alex Champandard, AiGameDev.com

AI
ARTIFICIAL INTELLIGENCE SUMMIT