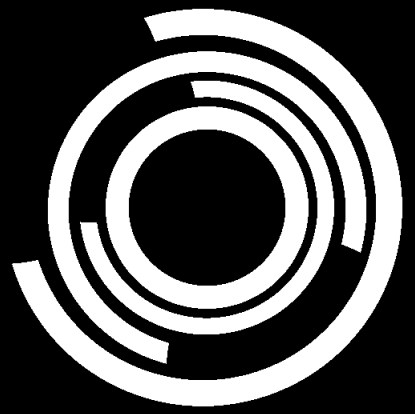


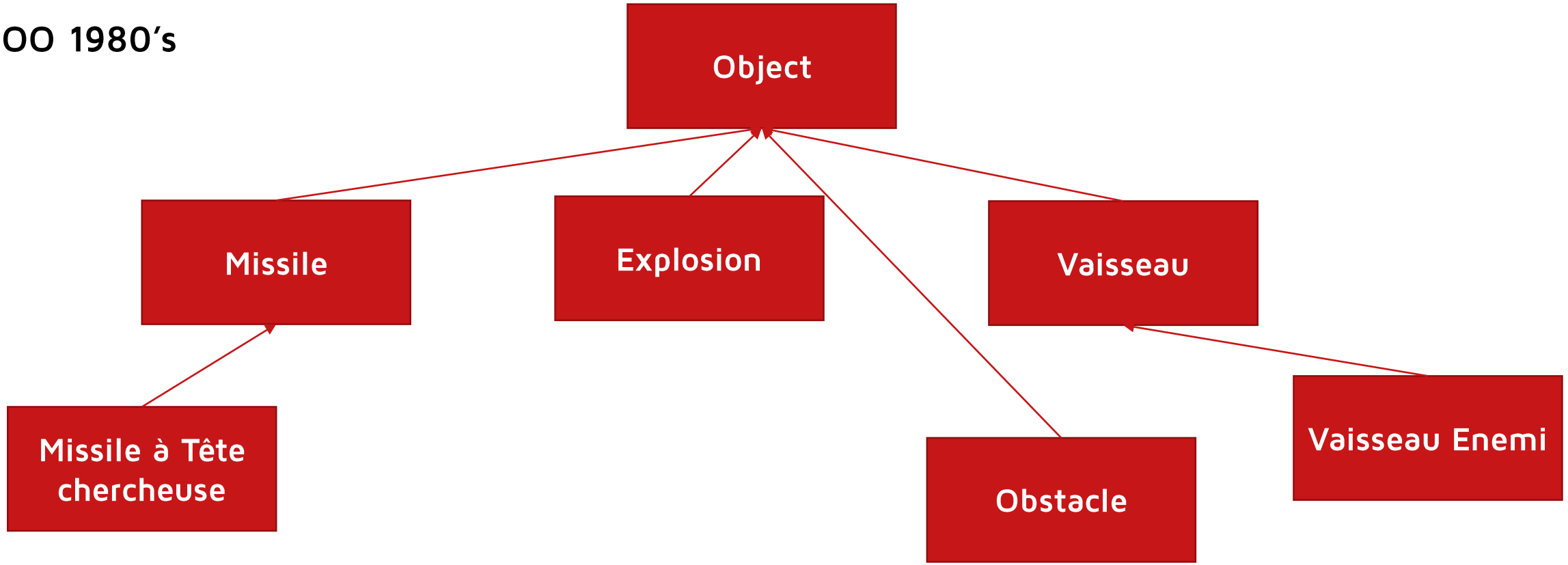
Architecture non-conventionelle

Désengagement de la POO

- **Problèmes**
  - Dans de nombreux domaines, l'acteur principal est la donnée
  - La génération des données et des comportements associés est délégué à des non-experts informaticiens mais experts métiers
  - Ex: le jeu vidéo
    - Les développeurs doivent fournir un cadre
    - Les créatifs génèrent du contenu pour ce cadre
    - Les créatifs ne codent pas
  - **Jeu vidéo, simulation, multi-agent = même combat**

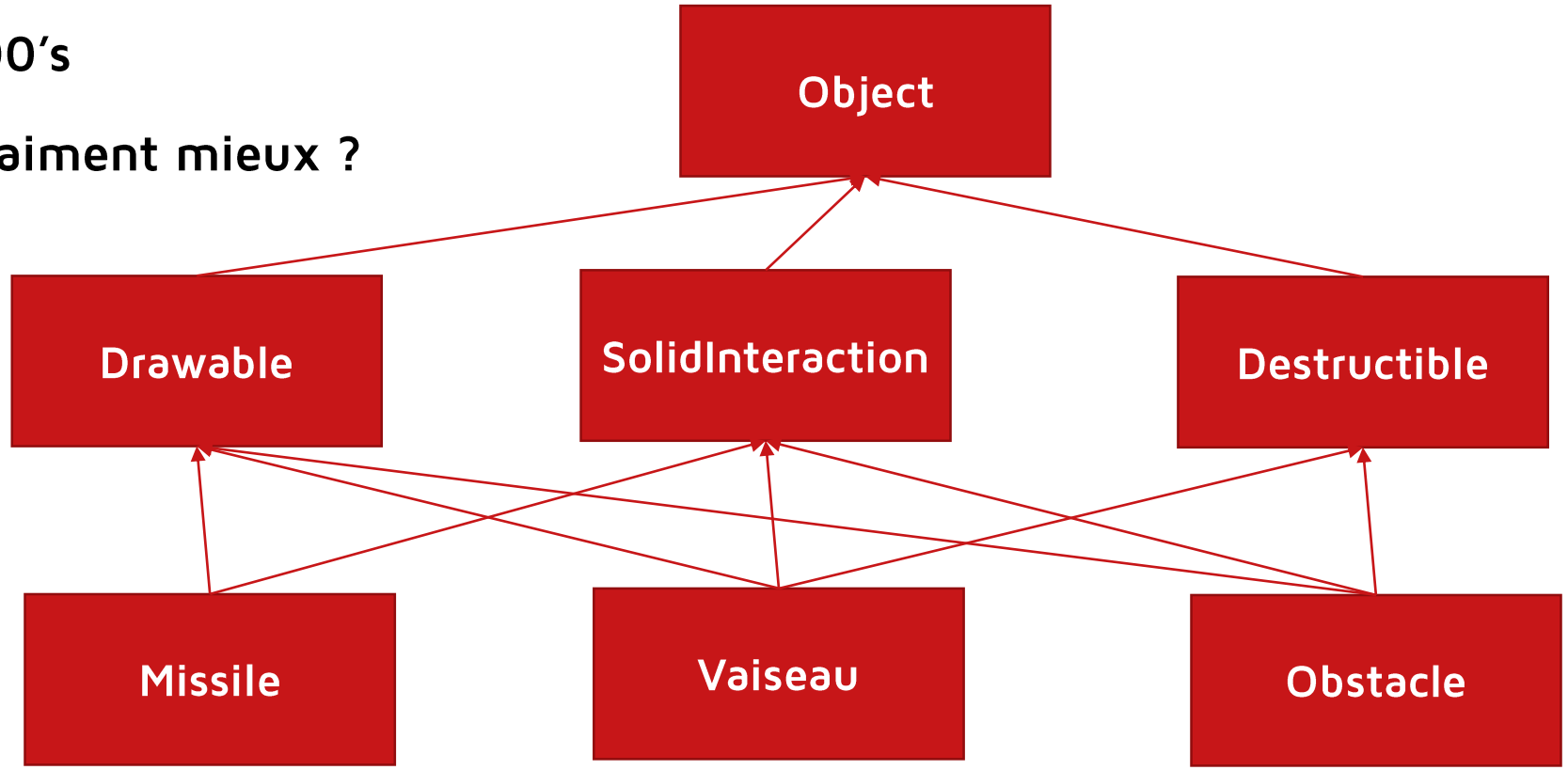
La POO n'est pas une panacée

- POO 1980's



La POO n'est pas une panacée

- POO 2000's
- Est-ce vraiment mieux ?



Reformulation du problème

- **C'est un problème de Base de données**
  - Une base de données recodée à chaque fois
  - Avec une structure rigide la ou un *schema* serait suffisant
  - Sans relation avec les données

- **Nouvel objectif**
  - Peut on recoder un ensemble de comportement polymorphe dirigé par les données ?
- **Le modèle Entity-Component-System**

Entity Component System

- **Une entité**
  - Objet trivial sans interface polymorphe
  - Représenté par un ID
  - Stockée en masse dans un conteneur adéquat

- **Un composant**
  - Structure de donnée contenant les informations nécessaire à l'exécution d'un comportement
  - Indépendant des entités
  - Stockés en masse dans un conteneur adéquat

Entity Component System

```
struct entity
{
  entity(std::uint64_t i) : id_(i) {}

  auto id() const { return id_;  }

  private:
    std::uint64_t id_;
};
```

Entity Component System

```
struct position
{
  float x, y;
};

struct apparence
{
  char format;
};
```

Relation Entity/Component

- **Ajouter un comportement à une entité**
  - ID = identifiant + bitmap des comportement supportés
  - Implémentation simpliste : bit mask
  - Implémentation plus fluide : 2 identifiant séparés
- **Pour N entités existantes, il existes N composants pré-alloués qui attendent d'être activé**
- **Le traitement des entités se fait**
  - de manière contiguë et monotone
  - Bonne cohérence des caches
  - Bonne performance

Entity Component System

```
struct entity
{
  static std::uint64_t root_id = 0;
  entity() : id_(root_id++), components_(0) {}

  auto id() const { return id_;  }

  template<typename Component>
  void supports()  { components_ |=  (1 << Component::id); }

  template<typename Component>
  bool is_supported() const { return components_ & (1 << Component::id); }

  private:
  std::uint64_t id_;
  std::uint64_t components_;
};
```

Entity Component System

```
struct position
{
  static constexpr std::uint64_t id = 0;
  float x, y;
};

struct apparence
{
  static constexpr std::uint64_t id = 1;
  char format;
};
```

Entity Component System

```
std::vector<entity> entities(10);
std::vector<position> positions(10);
std::vector<apparence> apparences(10);

// Activation de la position de l'entité 0
entities[0].supports<position>();

// MAJ de sa position
positions[0] = { 8, 4 };
```

Stockage et Système

- **Stockage et génération**
  - Les conteneurs d'entités et de comportements doivent être centralisé
  - Le traitement des comportements doit être customisables

- **Notion de système**
  - Filtre les entités selon leur comportements
  - Applique les transformations nécessaires en toute connaissance de cause

- **Points critiques:**
  - Conserver la contiguïté des objets
  - Minimiser les dépendances inter-systme

Système simplifié

```
struct perform_movement
{
  std::vector<entity> positions(10);

  void run(std::vector<entity> const& es)
  {
    for (std::size_t i = 0; i < es.size(); ++i)
    {
      if (es[i].is_supported<position>())
      {
        positions[i].x += 0.1;
        positions[i].y += 0.1;
      }
    }
  }
};
```

Conclusion

- **Détachement total de la POO**
  - Comportement externalisé
  - Surcout du polymorphe annihilé
- **Que reste-il à faire ?**
  - Générer les entités depuis des fichiers externes (JSON, XML)
  - Stockés les comportements sous forme externalisés (LUA, Python)
  - Simplifier la gestion (pb d'invalidations d'itérateurs)
- **N'hésiter pas tester :**
  - https://github.com/alecthomas/entityx
  - https://github.com/SRombauts/ecs