Using Metaprogrammed Functors to Implement Double-Dispatch for Collision Handling

Tiago H. C. Nobrega, Diego D. B. Carvalho, Aldo von Wangenheim

1LAPIX - Federal University of Santa Catarina - Florianópolis - Brazil
{tigarmo,diegodbc,awangenh}@inf.ufsc.br

Introduction

- Collision detection engines and frameworks are difficult to design and riddled with efficiency, modularity and cleanliness trade-offs.
- The management of multiple object types colliding in a scene can be aided by the double dispatch mechanism.
- We show how both object-oriented and generic notions can be used together to implement efficient, clean double dispatch.

Bounding Hierarchies with Double-Dispatch and Metaprogramming

- Bounding trees [1] are commonly used in collision detection. It is natural and simple to have each bounding tree class inherit from a base class filled with common methods.
- Alexandrescu [2] shows how to use this kind of hierarchy to efficiently implement double dispatch with a matrix of function pointers.
- We expand this idea with functors.

<table>
<thead>
<tr>
<th></th>
<th>AABBTree</th>
<th>OBBTree</th>
<th>SphereTree</th>
</tr>
</thead>
<tbody>
<tr>
<td>AABBTree</td>
<td>f1</td>
<td>f2</td>
<td>f3</td>
</tr>
<tr>
<td>OBBTree</td>
<td>f2</td>
<td>—</td>
<td>f4</td>
</tr>
<tr>
<td>SphereTree</td>
<td>f3</td>
<td>f4</td>
<td>—</td>
</tr>
</tbody>
</table>

- Some C++ metaprogramming tricks offload the chore of filling the matrix to the compiler.
- The functor templates can be specialized as needed—the compiler will find the right instantiation.

```cpp
template<class TreeA, class TreeB>
struct CollisionFunctor{
    bool operator()(const TreeA&, const TreeB&); //
};
```

Conclusions

- Initial benchmarks: at worst 15% overhead.
- The idea of generic functor matrices can also be applied to other dispatch cases.

References