(Snippets from) Building Game-Based Trainers with the Delta3D Game Manager

Curtiss Murphy
Alion Science and Technology
BMH Operation
cmmurphy@alionscience.com

Chris Osborn
Naval Postgraduate School
cwosborn@nps.edu

David Guthrie
Alion Science and Technology
BMH Operation
dguthrie@alionscience.com

VMASC Tutorial
July 28, 2006

TUTORIAL

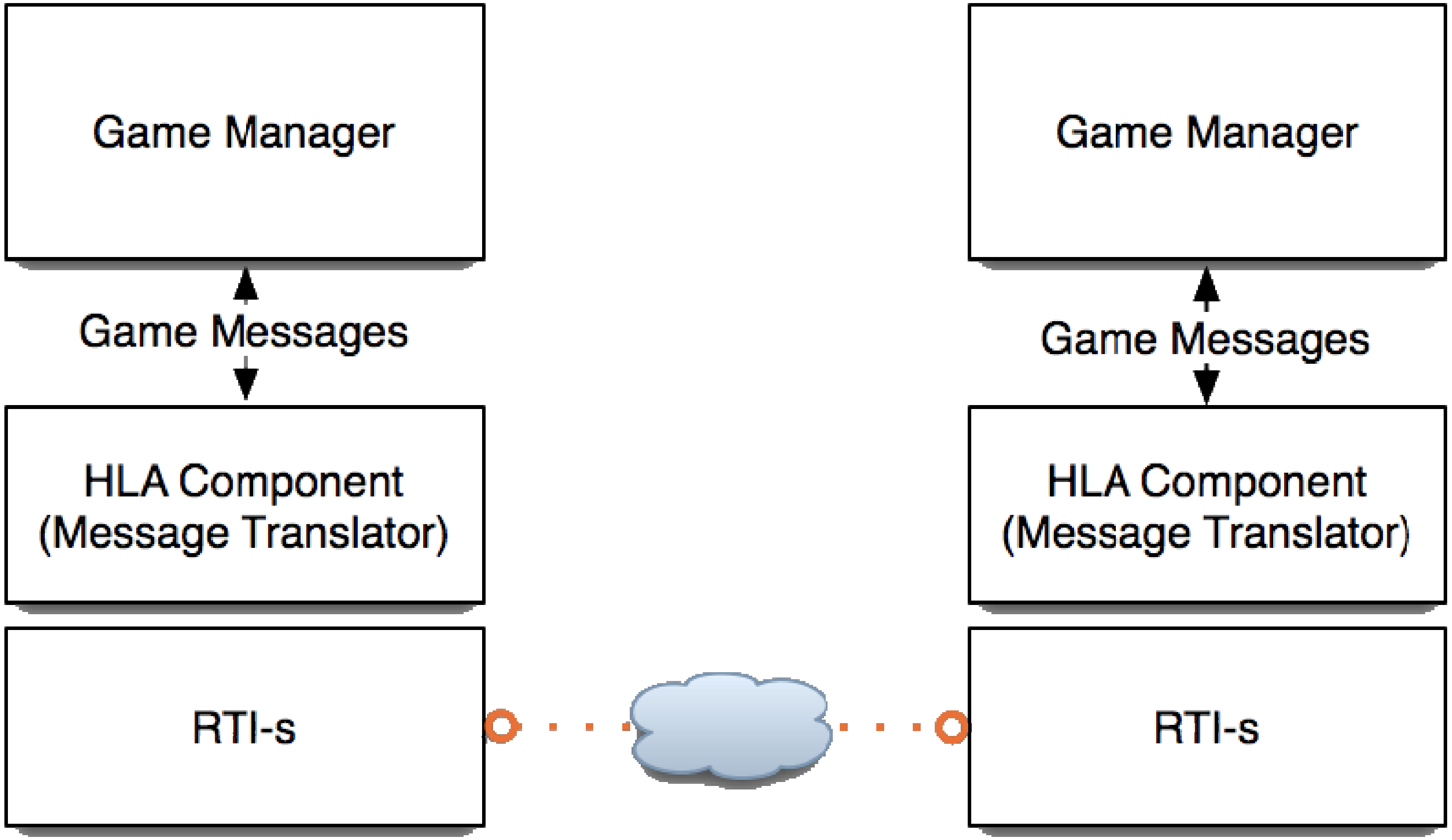
HLA

This presentation is an excerpt from the VMASC presentation, July 2006 at Old Dominion University in Suffolk, VA. Material is copyright Alion Science and Technology, 2006, 2007. First, read the I/ITSEC 2006 Delta3D Game Manager presentation, available from the Knowledge Base on the Delta3D website, www.delta3d.org.

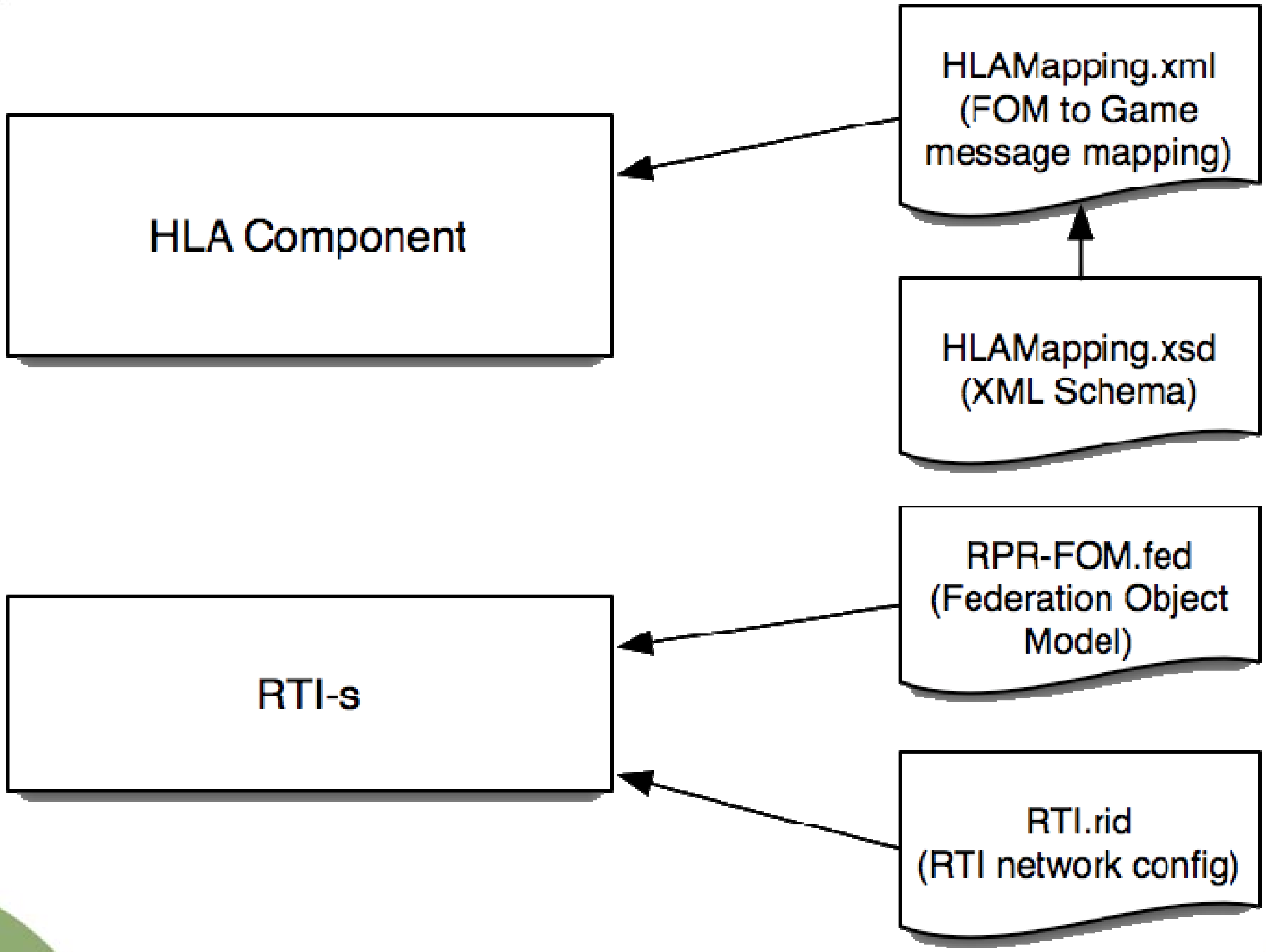
Part 6 - HLA Component

- Game Manager Component
- Configurable Message Translator
- Decouples HLA from Game Application Code
- Designed to support any federation object model with minimal coding
- Tested with several variants of RPR-FOM 1.0 and many recent releases of RTI-s

Part 6 - HLA Component



Part 6 -Configuration Files


HLA Component
HLAMapping.xml (FOM to Game message mapping)
HLAMapping.xsd (XML Schema)
RTI-s
RPR-FOM.fed (Federation Object Model)
RTI.rid (RTI network config)

Part 6 - HLA Networking

- Goal – Network multiple Tank Applications using the HLA Component.
- Our Tasks
  - Configure application for networking in general.
  - Create a second map to use with the second application instance.
  - Setup HLA Component and RTI configuration
  - Write HLA Mapping configuration
  - Tweak actor properties and update message passing

Part 6 - Rules Component

- Required for networked Game Manager applications.
- Defines the rules for sending and receiving network messages.
- Dispatches actor updates for published actors.
- Can be extended for more complicated game behavior
- Include dtGame/rulescomponent.h
- Snippet from MyGameEntryPoint.cpp

```
dtCore::RefPtr<dtGame::RulesComponent> rc
        = new dtGame::RulesComponent("Rules");
gameManager.AddComponent(*rc,
             dtGame::GameManager::ComponentPriority::NORMAL);
```

Part 6 - Publishing Actors

- A published actor is expected to be replicated over the network.
- There are three ways to publish an actor
  - The passing true for the isPublished parameter to GameManager::AddActor()
  - Calling GameManager::PublishActor() after the actor has been added
  - Setting the “Initial Ownership” property on a GameActor in STAGE to Server+Published.
- In our case, the Tank and the Blimps were set to Server+Published in the map.

Part 6 - Creating a new map

- This is simply to place the blimps and tank and different starting positions and give them different unique id’s.
- Typically, anything that needs to vary at runtime would be created in code.
- The new map is named “mapthree”. I’m not sure why.
- A command line option was added, “--mapName,” and handled in the Initalize method of MyGameEntryPoint.cpp. This allows picking the map without a recompile.

Part 6 - Add HLA Component

- Include

```
#include <dtHLAGM/hlacomponent.h>
#include <dtHLAGM/hlacomponentconfig.h>
```

- Create and Add Component

```
dtCore::RefPtr<dtHLAGM::HLAComponent> hlaComp =
        new dtHLAGM::HLAComponent("HLAComponent");
gameManager.AddComponent(*hlaComp,
        dtGame::GameManager::ComponentPriority::NORMAL);
```

- Config

```
dtHLAGM::HLAComponentConfig hlaCC;
…
hlaCC.LoadConfiguration(*hlaComp, mappingPath);
```

- Joint Federation

```
hlaComp->JoinFederationExecution("Tutorial", fedPath, "Tank Tutorial");
```

Part 6 - HLAMapping.xml

```
<header>
     <name>Tutorial</name>
     <description>Tutorial HLA mapping</description>
     <disEntityTypes>true</disEntityTypes>
             <disEntityTypeAttribute>EntityType</disEntityTypeAttribute>
     <author></author>
     <comment></comment>
     <copyright></copyright>
     <schemaVersion>1.0</schemaVersion>
 </header>

 <libraries>
   <actorLibrary>
     <name>TutorialLibrary</name>
     <version>1.0</version>
   </actorLibrary>
 </libraries>
```

Part 6 - HLAMapping.xml

```
<object name="BaseEntity">
  <abstract/>
  <entityIdAttributeName>EntityIdentifier</entityIdAttributeName>
  <attrToProp>
    <hlaName>DamageState</hlaName>
    <hlaDataType>UNSIGNED_INT_TYPE</hlaDataType>
    <gameName>Current Health</gameName>
    <gameDataType>INT</gameDataType>
  </attrToProp>
  <attrToProp>
    <hlaName>Orientation</hlaName>
    <hlaDataType>EULER_ANGLES_TYPE</hlaDataType>
    <gameName>Rotation</gameName>
    <gameDataType>VEC3</gameDataType>
  </attrToProp>
  <attrToProp>
    <hlaName>WorldLocation</hlaName>
    <hlaDataType>WORLD_COORDINATE_TYPE</hlaDataType>
    <gameName>Translation</gameName>
    <gameDataType>VEC3</gameDataType>
  </attrToProp>
</object>
```

Part 6 - HLAMapping.xml

```
    <object name="HoverTank" extends="GroundVehicle">
 <objectClass>BaseEntity.PhysicalEntity.Platform.GroundVehicle</objectClass>
   <actorType>MyActors.Tanks.Tank</actorType>
   <disEntityEnum>
      <kind>1</kind>
      <domain>1</domain>
      <country>222</country>
      <category>2</category>
      <subcategory>4</subcategory>
      <specific>3</specific>
      <extra>0</extra>
   </disEntityEnum>
   <attrToProp>
      <gameName>static mesh</gameName>
      <gameDataType>StaticMeshes</gameDataType>
      <default>StaticMeshes::bmp_hover.ive</default>
   </attrToProp>
</object>
```

Part 6 - HLAMapping.xml

```
  <object name="Blimp" extends="Aircraft">
 <objectClass>BaseEntity.PhysicalEntity.Platform.Aircraft</objectClass>
 <actorType>MyActors.Targets.Killable Target</actorType>
 <remoteOnly>false</remoteOnly>
 <disEntityEnum>
     <kind>1</kind>
     <domain>2</domain>
     <country>225</country>
     <category>1</category>
     <subcategory>9</subcategory>
     <specific>3</specific>
     <extra>0</extra>
 </disEntityEnum>
 <attrToProp>
     <gameName>static mesh</gameName>
     <gameDataType>StaticMeshes</gameDataType>
     <default>StaticMeshes::happyBlimp.ive</default>
 </attrToProp>
 <attrToProp>
     <gameName>Scale</gameName>
     <gameDataType>VEC3</gameDataType>
     <default>0.5 0.5 0.5</default>
 </attrToProp>
</object>
```

Part 6 - HLAMapping.xml

- More of the properties could have been sent out using a custom FOM.
- This includes the name of the resource used for the model, the shader state, etc.
- Using interactions, we could let the tanks target and destroy blimps on remote systems.
- dtGame has a DeadReckoningComponent. This allows for smooth movement of remote objects between updates. This requires a bit more setup including sending an enumeration over HLA about which algorithm to use.

Part 6 - Game Tweaks

- The blimps currently don’t actor updates about their state.
- The also need a property for their health so the remote system can see when to mark it damaged. They also assume a little too much about the way the health is being set.
- Blimps need to send an update when the scene is reset (R key).
- No changes need to be made to the Tank actor.

Part 6 - Blimp Health

```
void KillableTargetActor::SetCurrentHealth(int currentHealth)
{
   currentHealth = dtUtil::Max(currentHealth, 0);
   if( currentHealth == 0 && mCurrentHealth > currentHealth && mCurrentHealth != 0 )
   {
      mLargeExplosion->SetEnabled(true);
      SwitchVisitor switchVisitor("Destroyed");
      GetOSGNode()->accept(switchVisitor);

      mCurrentShaderName = "Normal";
      ApplyMyShader();
   }
   else if (currentHealth != 0 && mCurrentHealth > currentHealth)
   {
      mSmallExplosion->SetEnabled(true);
   }
   else if (currentHealth > 0 && mCurrentHealth == 0)
   {
      SwitchVisitor switchVisitor("Good");
      GetOSGNode()->accept(switchVisitor);
   }
   mCurrentHealth = currentHealth;

   GetGameActorProxy().NotifyFullActorUpdate();
}
```

Part 6 - Done!

RUN IT!

Part 6 - Bugs

- "R" has to be pressed in the first app run to make it send updates since the create messages were sent and lost before the second app started up.
- The Blimps need to send periodic updates to fix this.
- Delta3D 1.3 has a bug in outgoing HLA that had to be fixed for this tutorial. Here is the fix on line 1659 of hlacomponent.cpp

```
const AttributeType& hlaType
    = vectorIterator->GetHLAType();

+ if (hlaType == AttributeType::UNKNOWN)
+    continue;

size_t bufferSize;
char* buffer;
```

THE END

SimTime: 3397.81
Num Msgs: 17601
Last Msg: Actor Updated