XML-Based 3D Models for High Fidelity End Game Methodology

Jeff Lyons, Dr. David Fisher, Matt Kraus
Applied Research Associates, Inc.
Orlando, FL
jlyons@ara.com, dfisher@ara.com, mkraus@ara.com

ABSTRACT

As the military moves toward higher fidelity modeling for live and constructive training, they need more precise 3D models of target entities. This paper presents an approach for using 3D target models in a high fidelity end game methodology. ("End game" in this paper refers to the end of an engagement; i.e., a munition impacting its target.) The methodology uses the models to determine if a player was hit in an engagement, and if so, where the impact occurred.

Military training and testing systems are one of many possible applications for this methodology. In future live training and testing domains, each player unit holds the 3D model of the target it represents. This allows the embedded unit to run the algorithm and advise the trainee if he/she was hit by an engagement and if so, approximately where. Using high fidelity models results in less false hits and false misses, avoiding negative training. Other applications of this strategy include sensor system evaluation and calculation of visual center of mass.

This paper will discuss the basics of the methodology including how 3D information is stored on the player unit, inputs required to calculate hit location, orienting the model for delivery, and the hit location calculation. We will present an implementation of the algorithm and strategies to optimize processing and memory usage.

ABOUT THE AUTHORS

Jeff Lyons is a senior engineer at Applied Research Associates. He has a Bachelor of Science degree in mechanical engineering from Florida State University and a Masters in mechanical engineering from the Massachusetts Institute of Technology. Jeff's experience spans the mechanical engineering and software engineering disciplines. He has experience in structural analysis, CAD design, manufacturing support, and test support. He also has worked extensively in simulation and software design. He has worked on tasks related to military training and testing systems.

Dr. David L. Fisher is a senior scientist for Applied Research Associates. Since receiving his doctorate in physics from the University of Texas at Austin in 1995, David has worked in the areas of theoretical plasma physics, laser-plasma interactions, laser wakefield acceleration, detection of explosive materials, millimeter wave sensors and applications, an Internet business, and systems and software engineering. David is presently supporting the development of area weapon effects models. David has 22 professional publications and 2 patents (1 pending).

Matt Kraus is a principal scientist at Applied Research Associates. He has a Bachelor of Science degree in computer science from Western Michigan University and a Master of Science degree in simulation, modeling, and analysis from the University of Central Florida. His research interests are in the areas of distributed computing, artificial intelligence, and computer graphics.

XML-Based 3D Models for High Fidelity End Game Algorithm

Jeff Lyons, Dr. David Fisher, Matt Kraus
Applied Research Associates, Inc.
Orlando, FL
jlyons@ara.com, dfisher@ara.com, mkraus@ara.com

INTRODUCTION

Force-on-force live training systems (MILES, 2006) typically assume that a shooter aimed and shot at a target's visual center of mass (AMSAA, 2004). Historically, sensor systems were too inaccurate to give a precise aim point. Note in Figure 1 that the shooter is aiming at a point (the true WPD aim point) well below the visual center of mass.

Technological advances in orientation sensors have made it possible to more accurately determine the weapon pointing direction (WPD). GPS systems can report the shooter and target position. From this information we can calculate the simulated munition's trajectory. We can create an "electronic bullet", a message that contains engagement parameters necessary to determine the results of the engagement. The information in the electronic bullet can also be used to calculate the measured WPD aim point shown in Figure 1.

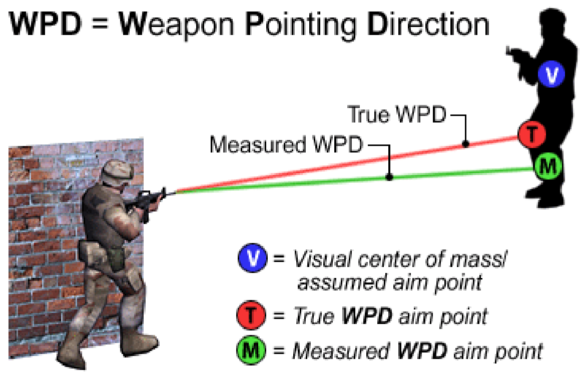

WPD = Weapon Pointing Direction
True WPD
Measured WPD
V
T
M
V = Visual center of mass/ assumed aim point
T = True WPD aim point
M = Measured WPD aim point

Figure 1. Current systems assume the shooter accurately aimed at the visual center of mass (point V). Our proposed methodology uses the measured weapon WPD aim point (point M) to determine the engagement result.

The *vertical target plane*, or *VTP*, is the plane passing through the target's visual center of mass that is normal to the vector between the shooter and the target. We will refer to the intersection of the simulated munition with the vertical target plane as the *simulated intersection point*.

The simulated intersection point is more realistic when using a more accurately measured WPD aim point. We can use this more realistic simulated intersection point to determine more precisely if a target was hit. If the target was hit we can also report what part of the target was hit (e.g. head or leg for dismounted infantry, turret or hull for a tank). This methodology uses a 3-dimensional model of the target to make that assessment.

A higher fidelity end game methodology would benefit many applications. Vulnerability calculations can use the additional information about what part of the target was hit to perform a higher fidelity vulnerability assessment, resulting in more accurate engagement results. Training systems can use this information for health assessment and realistic medical training. The trainee's confidence in training is enhanced by the higher fidelity end game assessment. Also, with a more accurate target model, we can make a more accurate estimate of the target's visual center of mass and visual mass area.

Sensor system evaluation is another potential application of this methodology. Using these methods with 3D target models, simulations could evaluate a sensor system's algorithms against targets of various sizes and geometries. Setting up laboratory tests for these various targets would be expensive and impractical. This methodology could be used to simulate these tests without laboratory setup. This application is not further explored in this paper.

BACKGROUND

Aim Point Error

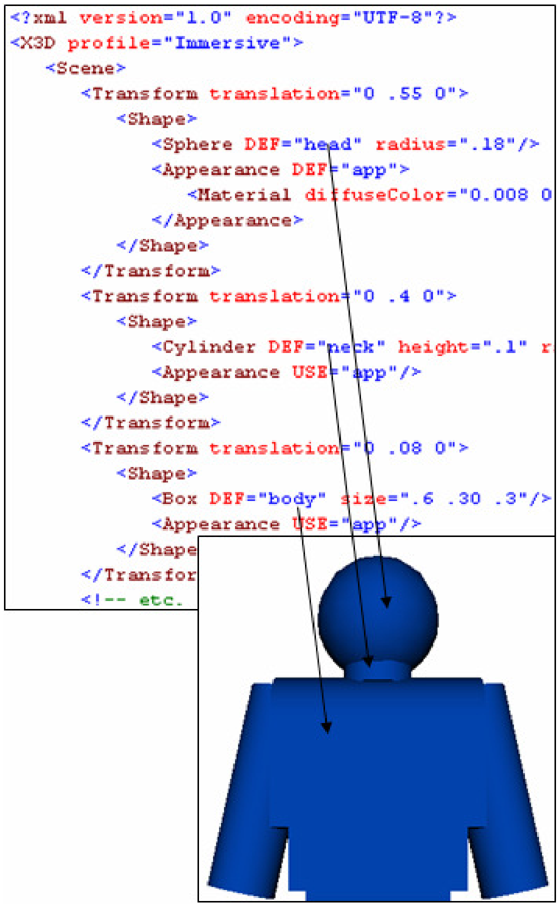
Note that in Figure 1 the measured WPD aim point is a closer approximation to true WPD aim point than the visual center of mass. For this to truly be the case, the sensors measuring position and orientation must meet certain accuracy criteria. These are discussed in "Calculating Error Tradeoffs in Weapon Simulation for Live Training" (Hall, 2006). In some cases, one may have some confidence in the measured WPD aim point, but not enough confidence to trust it entirely. In this case, an averaging or hybrid methodology can be used. In this paper we assume some intersection point with the vertical target plane has been chosen. It can be the measured WPD aim point, the visual center of mass, or some point in between.

3-Dimensional Geometric Formats

We considered several factors in choosing a 3D format for use with our methodology. This methodology needs a polygonal representation of the target. We read the model file at initialization and store the polygons. Therefore we have no need for the model file after initialization. Thus our requirements are a format that is easy to parse and easy to build polygons from. We prefer an ASCII based XML format as these are simplest to parse and build hierarchies from. We also prefer a format which has a pre-existing comprehensive library of military models.

X3D is a common format with all the features necessary to support this methodology (Web 3D Consortium, 2006). The format is an extensible, XML-based format for representing 3-dimensional objects. It is the successor to the popular VRML format. There is a wide variety of military equipment models available in X3D. The Navy SAVAGE project has 972 military models available online (Brutzman, 2006). The Army Model Exchange also has a comprehensive library of 674 military models (Department of Defense, 2006).

Figure 2 shows an example of the X3D format.



```
<?xml version="1.0" encoding="UTF-8"?>
<X3D profile="Immersive">
   <Scene>
      <Transform translation="0 .55 0">
         <Shape>
            <Sphere DEF="head" radius=".18"/>
            <Appearance DEF="app">
               <Material diffuseColor="0.008 0
            </Appearance>
         </Shape>
      </Transform>
      <Transform translation="0 .4 0">
         <Shape>
            <Cylinder DEF="neck" height=".1" r
            <Appearance USE="app"/>
         </Shape>
      </Transform>
      <Transform translation="0 .08 0">
         <Shape>
            <Box DEF="body" size=".6 .30 .3"/>
            <Appearance USE="app"/>
         </Shape
      </Transfor
      <!-- etc.
```

Figure 2. Sample X3D file. X3D uses XML syntax with nodes for appearance, 3D primitives, rotational and translational transforms, and more. The model shown here describes a human head, neck, and body.

The shapes are defined in XML style blocks. The hierarchal nature allows grouping and nesting of shapes to facilitate association and geometric transforms. Note also the material properties block. Since the format is extensible, this could be used to define any material property imaginable. (Material properties are not currently used by this methodology but could be used for vulnerability calculations.)

METHODOLOGY DESCRIPTION

In this section we describe the methodology used to determine a munition's impact on a target using an X3D model of the target's geometry. We will describe the general software architecture, model initialization, and engagement processing. We will also describe optimization strategies and visual center of mass calculation.

Methodology Architecture

The methodology architecture uses inheritance and object containment for efficiency, maintainability, and flexibility. The architecture is illustrated in Figure 3.

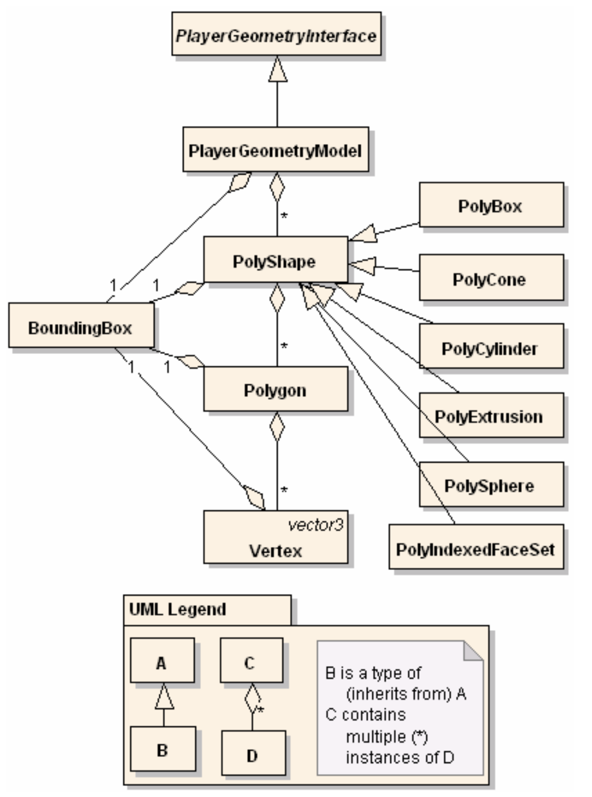


Figure 3. Player geometry class diagram. A PlayerGeometryModel is a collection of PolyShapes. PolyShapes are a collection of Polygons, which are a collection of Vertices. Each hierarchal level has a BoundingBox.

These classes contain the polygons that define a target's 3-dimensional geometry. The PlayerGeometryInterface provides a common base class so others wishing to define their own geometry model may do so with minimal impact to the software. The PlayerGeometryModel realizes this interface. It contains multiple PolyShapes. These shapes can contain several Polygons. A Polygon can contain multiple Vertices. Note that all levels in this hierarchy contain a BoundingBox. This box serves as a filter for engagement processing (see optimization strategies section).

There are six types of PolyShapes. They of course all have different shapes, but are all built from the collection of Polygons.

Model Initialization

At initialization we build the player geometry model from the X3D file. Initialization is illustrated in Figure 4.

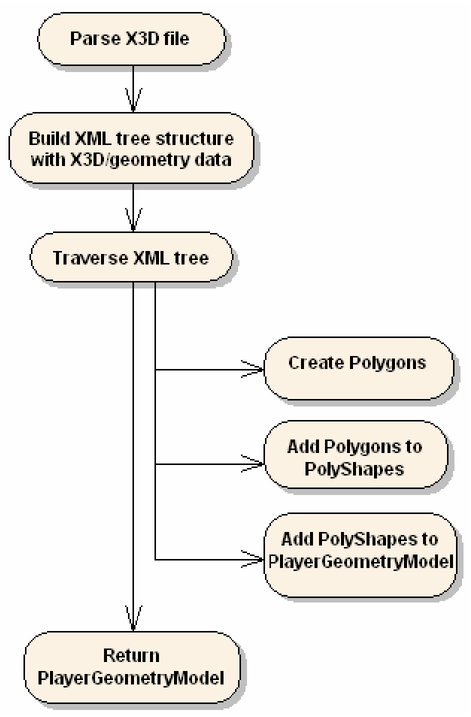

Parse X3D file
Build XML tree structure with X3D/geometry data
Traverse XML tree
Create Polygons
Add Polygons to PolyShapes
Add PolyShapes to PlayerGeometryModel
Return PlayerGeometryModel

Figure 4. Software Initialization. We parse the X3D file, build an XML tree from the X3D model, then traverse the tree, building the player with polygons and shapes.

We parse the X3D file using a standard third party XML parser (Apache Software Foundation, 2006). The XML parser outputs the node model tree defined in the file. We then traverse the tree, creating polygons as we go from the node information. The polygons are added to the PolyShapes, which are in turn added to the PlayerGeometryModel. At the end of this process, the PlayerGeometryModel is complete and we no longer need the information parsed from the X3D file.

Methodology Processing

This section describes the processing of the engagement. These steps determine if the target was hit and if so, where. The process is outlined in Figure 5.

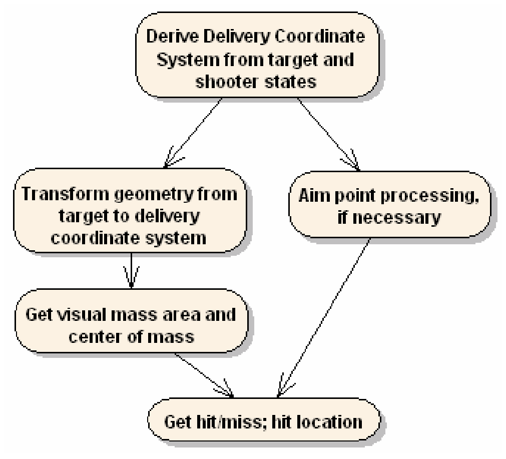

Derive Delivery Coordinate System from target and shooter states
Transform geometry from target to delivery coordinate system
Aim point processing, if necessary
Get visual mass area and center of mass
Get hit/miss; hit location

Figure 5. We calculate hit or miss via this five step engagement process.

Firstly we derive a coordinate system convenient for engagement processing. We transform the target geometry to this coordinate system then calculate the visual mass area and center of mass. In parallel to these steps, we process the aim point to get its intersection with the vertical target plane. (We will not describe this step since it is trivial and the aim point could be pre-processed making this step unnecessary.) Finally, we determine if the target was hit. If it was hit, we find the impact point.

We derive the delivery coordinate system from the inputs. Defining our geometry in the delivery coordinate system makes processing simpler and more efficient. The following figure illustrates this coordinate system.

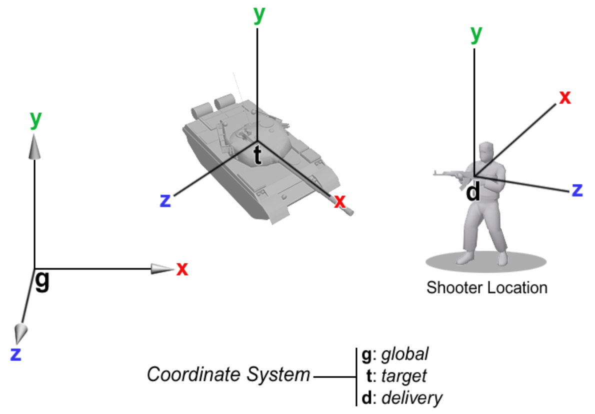


Figure 6. The delivery coordinate system puts all the relevant geometry into an XY plane, turning a 3D problem into a 2D problem.

The origin of the coordinate system is at the shooter's weapon location. We define the Z axis of the delivery coordinate system as the vector from the target's visual center of mass to the shooter's weapon location. The X axis is defined by being orthogonal to Z and parallel to the ground plane (global XZ plane). The Y axis completes a right-handed coordinate system.

Next, we transform the player model into delivery coordinate system coordinates. The shooter is viewing the vertical target plane, which is parallel to the delivery coordinate system's XY plane. This allows us to ignore the Z coordinate of the geometry and assess the hit based purely on the X and Y coordinates, turning a 3D problem into a 2D problem.

Finally, we determine if the simulated munition impacted the target. (The intersection point of the munition with the vertical target plane must be provided as an input.) If we determine a hit occurred, we will in turn determine the impact location. We traverse the model tree, iterating through the shapes and polygons. We potentially check each polygon to see if the plane intersection point is inside. (Typically, each polygon is not checked. See optimization strategies section on bounding box filtering below.) Because we transformed the geometry, we can ignore the polygons' Z coordinate and use a 2D polygon check to see if the point is inside.

We use the Jordan Curve Theorem to determine if the intersection point is inside the polygon (Haines, 1994). Given a point, draw a line from it to the polygon's furthest vertex from that point. The theorem states that if the line crosses polygon edges an odd number of times, the point is inside the polygon; an even number and it is outside. (Count a vertex intersection as one.)

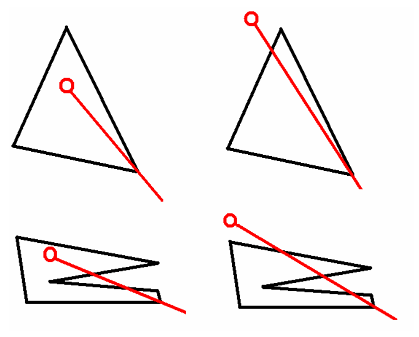


Figure 7. Both polygons on the left have points inside and the line crosses the edges an odd number of times. Both polygons on the right have points outside and the line crosses the edges an even number of times.

Figure 7 illustrates this principle. For each point (shown as circles) a line is drawn from it through the polygon's furthest vertex. For trivial polygons with no concave edges like the triangles at the top, the line crosses the edges once when inside (odd number), and twice when outside (even number). For more complex shapes like the two on the bottom, points might cross the edges more than once or twice, but the principle still holds. The one on the bottom left crosses the edges 3 times (odd number). The outside point on the bottom right crosses the edges 4 times (even number).

If we find an impact, we save the point along with its distance to the target. We continue to search for impact points. When another is found, the one closest to the target is saved and the other discarded. This ensures we will have the impact point that will be reached by the munition first. If no impact point is found, the simulated munition did not hit the target.

Optimization Strategies

We employed various strategies to ensure the software runs in real time, regardless of model fidelity level. Each element in the player geometry hierarchy has a bounding box starting with the entire player geometry model (see Figure 3). These bounding box checks are done at all levels in the hierarchy to ensure the expensive polygon check previously described will only be done when the munition is in the neighborhood of the geometry to be checked. The following figure illustrates the bounding boxes.

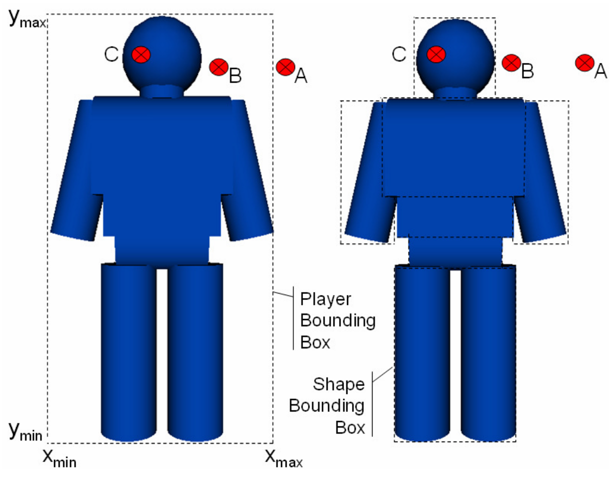


Figure 8. Bounding boxes at different hierarchal levels. The figure at left shows the bounding box around the entire player. The one at the right shows bounding boxes around the shapes. Polygons and vertices also have bounding boxes (not shown).

The points A, B, and C in Figure 8 illustrate the use of bounding boxes. A shot at point A is outside the bounding box of the entire player model, and immediately returns a miss with no further processing. Point B is inside the player model boundary, thus we iterate through the shapes and look for a hit in each. However, each check immediately returns as point B is outside the bounding box for each. Point C is similar, except for one shape (the head and neck). We iterate through this shape's polygons searching for a hit. The polygons also have bounding boxes (not shown). Thus we only do the computationally expensive polygon check for a few polygons instead of the hundreds in this model.

Another strategy we use to ensure computational efficiency is results caching. This refers to saving then reusing results or other calculated parameters from the previous engagement if the engagement input parameters are similar enough. We save several parameters from the previous engagement. Based on the "level of similarity" shown in Table 1, we can reuse some of these parameters, saving the computational time we would have spent recalculating them. These similarity levels are shown in the following table.

Table 1. Similarity Levels. How similar is the current engagement to the previous. Used to determine how much we can reuse cached results.

Similarity Level	Description
All Same	✓ The target and shooter locations are the same ✓ The target orientation is the same ✓ The weapon pointing direction is the same
Locations and orientation same	✓ The target and shooter locations are the same ✓ The target orientation is the same
Locations same	✓ The target and shooter locations are the same
None same	✓ Nothing is the same

To illustrate results caching, imagine a target and shooter who are in set positions. The shooter is shooting a mounted weapon. From shot to shot, the locations, target orientation, and weapon pointing direction do not change. (Burst fire weapon engagements are one example of when this might occur.) Therefore, many steps in the engagement process do not need to be redone, including the expensive step of transforming the target geometry to the delivery plane. How much can be reused depends on how similar the engagements are.

Visual Center of Mass and Mass Area

This methodology provides a convenient method for calculating the visual center of mass and visual mass area. The visual center of mass is the area center of mass from the shooter's perspective in the vertical target plane. The visual mass area is the area of the target profile in the plane.

These values are not necessary for determining hit if the munition intersection with the vertical target plane is provided. However, hybrid approaches may use an average of the visual center of mass and the measured weapon pointing direction aimpoint. If these values are desired, this methodology can provide them.

To calculate visual center of mass and visual mass area, we divide the player bounding box into a grid (left image, Figure 9). We then check the center point of each grid cell for geometry (right image, Figure 9). To check for geometry we use the same polygon algorithm previously described for checking target/munition impact. If there is geometry, we increase the count of visual mass area by the cell area, and add the point to a running average to get the center of mass.

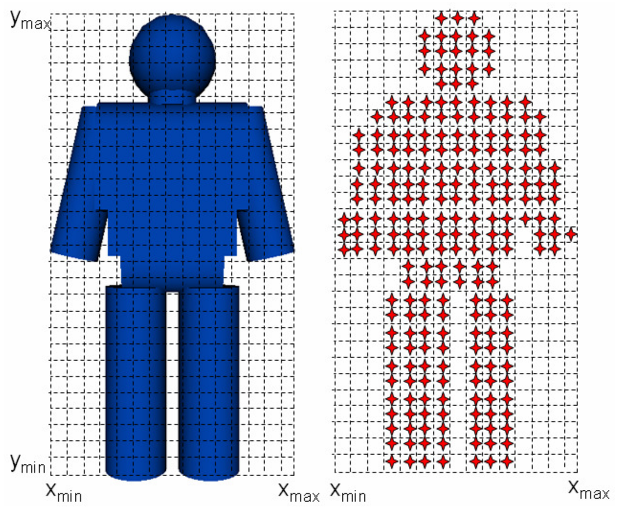


Figure 9. To determine visual center of mass and visual mass area, we divide the bounding box into a grid (figure on left), then iterate through the grid cells checking for geometry in each (figure on right).

CASE STUDY RESULTS

We performed tests to assess the processing time used by the algorithm. The tests were performed on three target models of varying fidelity level.

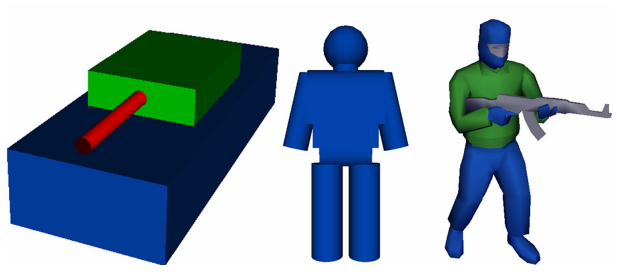


Figure 10. Models used in timing tests. The low fidelity tank at left has 24 polygons, the simple soldier in the middle has 128 polygons, and the detailed soldier at right has 1214 polygons.

The tests also evaluated the effectiveness of the optimization strategies. We ran tests with and without results caching. We also ran tests with and without the bounding box filtering technique. The tests were run on a laptop machine with the following specifications.

Laptop Specs
2.26 GHz Intel Pentium processor
2.00 GB RAM

We also tested on an embedded processor for comparison. Fewer tests were run on this processor, but those that we ran had nearly identical results to the laptop results. These are not shown below since they were so similar.

Embedded Processor Specs
500 MHz AMD Geode GX 533
500 MB RAM

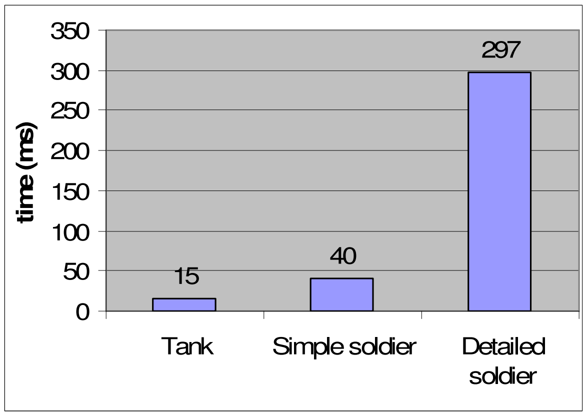


Figure 11. Processing time with different models. Bounding box optimization was used, but no results caching.

Figure 11 shows processing times per engagement (with bounding box optimization) were 15, 40, and 297 milliseconds for low, medium, and high fidelity models respectively. For most applications, a processing time of 40 milliseconds per engagement would be acceptable. The detailed model had a processing time of nearly 300 milliseconds. Whether or not this would be acceptable depends on the application. This would be fine for most live training exercises assuming there is very little further processing that must occur. If further processing is required, a less detailed model should be used.

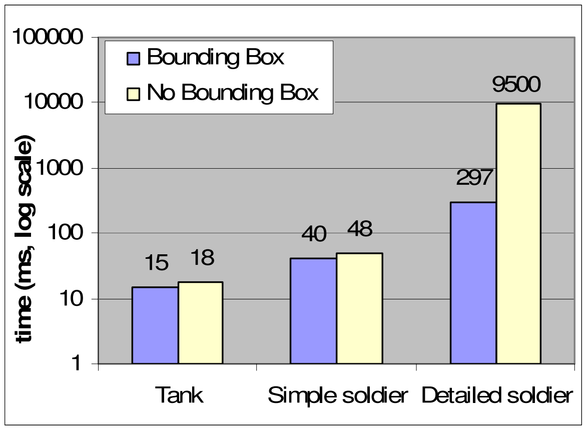


Figure 12. Processing time with and without the bounding box optimization. Note it is very important for the detailed model.

Figure 12 demonstrates the importance of the bounding box optimization, particularly for larger models. This result shows that bounding boxes are a critical part of the methodology.

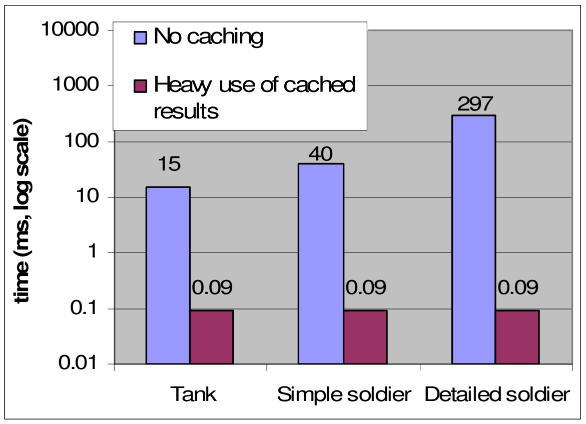


Figure 13. Processing time with and without results caching. Note reusing parameters greatly improves efficiency.

Figure 13 shows that results caching can greatly improve your efficiency. Note however that this assumes the engagement is nearly identical to the previous. This will happen occasionally, but not frequently. Occasionally there will also be a medium level of reuse, when the engagement is somewhat similar to the previous. Therefore, this strategy is helpful to reduce processing time, but cannot be relied on consistently due to its dependence on scenario events.

CONCLUSION

A high fidelity end game methodology using 3D target models is now feasible due to technological advances. This methodology can more accurately assess the results of an engagement, for both munition delivery accuracy and damage assessment.

This paper demonstrated that an implementation of this methodology can be sufficiently optimized for real-time live training systems. For low to medium fidelity target models, the processing time was acceptable (less than 100 milliseconds). For high fidelity target models, the processing time was higher (about 300 milliseconds). Whether or not this is acceptable would depend on the application.

Further work in this area could be done. This methodology could be extended to other geometric formats. Other optimization techniques could be explored to improve processing time, particularly for high fidelity models. Also, the results of this methodology could be applied to physics-based models to give even higher fidelity engagement results.

ACKNOWLEDGEMENTS

Thanks to Mary Ann Pigora for her reviews and insightful comments.

REFERENCES

AMSAA. (2004). *Physical Model Knowledge Acquisition Document, Vulnerability from Direct Fire Weapons for Ground Vehicles*, document number KEMA070001.

Apache Software Foundation. (2006). *Xerces*; Validating XML parser located at http://xml.apache.org/xerces-c/

Brutzman, D. (2006). *SAVAGE Project*; Free repository of X3D Models from Navy Source located at http://web.nps.navy.mil/~brutzman/Savage/

Department of Defense. (2006). *Army Online Model Exchange*, paid subscription required, located at https://modelexchange.army.mil

Haines, E. (1994). Point in Polygon Strategies. *Graphics Gems IV*, pages 24-46.

Hall, R., & Janisz, M. (2006). Calculating Error Tradeoffs in Weapon Simulation for Live Training. *IITSEC 2006*, Paper number 2515.

MILES. (2006). Multiple Integrated Laser Engagement System. located at http://www.peostri.army.mil/PRODUCTS/MILES/

Web 3D Consortium. (2006). *X3D*; XML based open file format for representing 3D scenes. Located at: http://www.web3d.org/