A Comprehensive Interactive Approach for Training Simulation of Equipment Decontamination

Sijiang Liu, Yongshi Jiang, Yiping Yang
Integrate Information System Research Center
Institute of Automation, Chinese Academy of Science
Beijing, China
sijiang.liu@ia.ac.cn

Liang Yu, Xiaobing Geng
Department of Nuclear Defense
Institute of Chemical Defense
Beijing, China

***Abstract*—To improve effectiveness of simulation training of equipment decontamination, it's essential to emulate and process interactions in virtual scenes accurately, wishing to produce the same results as in reality. So we firstly build mathematical and attribute models for simulation targets, and then propose a processing algorithm based on those models to dispose various interactions and mark down critical changes. Finally an evaluation method is carefully chosen to score virtual trainings. The whole three steps constitute an original approach for training simulation of equipment decontamination, which features comprehensive interaction handling and incorporate solution from training to assessment. Tests have shown that our approach not only conducts virtual decontaminating as true as in real world, but also evaluate it automatically with both quantitative result and detailed text records. Besides the example equipment used in this article, the approach is also applicable to others, which reveals excellent generality. All these advantages make the simulation substitute for real training more reasonably and convincingly.**

Keywords-training simulation; equipment decontamination; modeling; processing algorithm; evaluation

I. INTRODUCTION

It's a trend to bring virtual reality technology in the field of military equipment training, like equipment decontamination, which could break through limiting factors such as time, region, personnel, environment, resource condition, etc.

Most doctrines and programs concerned with military affairs are not published for reasons of confidentiality, but related researches that aim at managing civilian biochemical warfare casualties are more common. Reference [1] systematically introduces four main categories of modern medical simulation. Their researches are mostly base on human model simulators, thus the above restrictions are inevitable. Compared to that, [2] supplies a computer-based virtual reality system for learning and practicing critical care medicine, which is very useful and convenient. Perhaps the only flaw is that interactions are performed by selecting actions from a pull-down menu in user interface of the program, and this kind of state-transition interaction is the most often choice in a educational or training VR system.

Command and Engineering Institute of Chemical Defense first apply VR technique to anti-chemical field[3][4][5][6]. Their simulation training system for decontamination vehicle is such a hardware-in-the-loop system containing functions like training configurations, visual and sound effects, interactive manipulation, assessment and so on[4][5]. However in detail, it is not impeccable, especially in equipment decontamination subsystem. The problem lies in that: (1) models for simulation targets are very coarse, so (2) the system could only emulate simple interactions, providing incomplete operation records, and (3) as a result there is no enough information for training evaluation as in reality.

In order to improve effectiveness of simulation training of equipment decontamination, the virtual system must cope with any operation that could happen in real world and engender certain results. Thus complete and automatic evaluation could be executed based on those results.

In this paper we present an original approach to achieve the above goal. The approach consists of three steps and the first two are core parts, which are more relevant to graphics, hence we will focus on them in this article.

To begin with, simulation targets should be abstracted with two kinds of models created. Mathematical models describe the targets' geometry, which are used for collision detection. Attribute models display targets' states related to decontamination, and states updating reveals effects caused by interactions.

Then a processing algorithm is proposed that utilize existed models to control emulation progress. The algorithm is characterized by the following set of features:

- Integrated into graphic rendering loop. That means the algorithm is synchronous with virtual scene. Corresponding operation will be executed as soon as some event is triggered in virtual scenes.
- Smooth, optimized processing in real time. Processing is accelerated for optimizing in collision detection and other parts, so it is rapid enough to keep the frame rate up to 30fps.

978-1-4577-0536-6/11/$26.00 ©2011 IEEE

- Comprehensive handling with almost all kinds of situations. The algorithm has considered nearly all possible manipulations, no matter right or wrong in reality and would deal with them as expected.

Finally when training is over, it will be scored automatically. The evaluation method we choose is analytic hierarchy process (AHP), which conforms to practical assessment pattern.

The following contents of this article are arranged as this. In section 2, we briefly explain conceptions related to equipment decontamination, point out difficulties in simulation and our goals. Section 3 provides modeling methods for simulation targets. Then processing algorithm is discussed in section 4. We verify our approach with results from designed tests in section 5 and conclude our paper in the last section.

II. RELATED CONCEPTIONS

Equipment decontamination means measures for cleaning equipment when it's been contaminated[7].

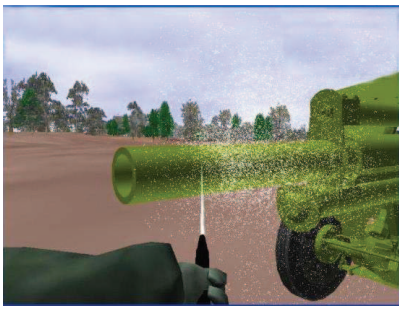


Figure 1. Typical 3D scene in simulation

The typical scene in our simulation is shown as Fig. 1, decontaminating personnel holding water-cannon to douche equipment in decontaminating field. Here we use a keyboard to move the virtual character and a mouse to jet the spout and change its orientation.

In practice, equipment is divided into several components that should be douched orderly. On components there are so called sensitive areas, which must be washed in decontamination. These areas are on the surface of components, usually in forms of points, segment or curve. When douching them, one should keep water spout hitting at the point, or washing from one end of the segment or curve to the other end without pause or deviation. Foremost, a component will not be considered to be decontaminated completely unless all sensitive areas on it have been douched. Additionally, length, angle and moving speed of the spout should be taken into account in assessment.

Since interactions are totally open, i.e. any operation may happen at any time, how to save previous processing states and swiftly convert to current interaction are main difficulties. So our goal is to present an approach that could comprehensively monitor and process decontaminating manipulations, recording needed information such as processing sequence, state changes of sensitive areas, state of waterspout and other things to support assessment.

III. MODELING

Obviously in Fig. 1, the objects mostly related to decontamination are water spout and equipment. Here we use artillery as an example to show how to model for it. But our method is applicable to other equipment.

A. Water Spout

Considering that decontaminating personnel use water-cannon within a short distance, it's reasonable to abstract water spout as a radial, i.e. hitting the artillery immediately without bend.

1) Mathematical Model

The model equation can be expressed as this:

$$\mathbf{l} = \mathbf{p} + t\mathbf{d} \tag{1}$$

Here vector $\mathbf{l}$ represents the whole radial, $\mathbf{p}$ is the starting point, and $\mathbf{d}$ is the direction of the radial. Parameter t determines extension of the radial, and $t \geq 0$.

2) Attribute Model

According to (1), there are two attributes:

a) *Starting point* $\mathbf{p}$.

b) *Direction* $\mathbf{d}$.

B. Artillery

Since equipment is factitiously partitioned into components that need to be decontaminated in order, it's better to model for components rather then entire artillery. The former choice would reduce the difficulty of modeling but improve the reusability of models.

1) Mathematical Model

At component level, math models are mainly in three forms.

First is the convex quadrilateral, which can be expressed as following:

$$Ax + By + Cz + D = 0 \tag{2}$$

$$\left(\overrightarrow{PV}_n \times \overrightarrow{PV}_{(n+1)\bmod 4}\right) \cdot \mathbf{n} > 0 \tag{3}$$

Equation (2) indicates a plane in 3d space, and $P = (x, y, z)$ is a point on the plane. But the plane is restricted to a convex quadrilateral area by (3). As Fig. 2 (a) and (b) shown, V_n, the vertices of the convex quadrilateral, are arranged counterclockwise with subscript n increasing($n = 0, 1, 2, 3$). Vector $\mathbf{n}$ is the normal of the plane according to right-hand rule. Only if (3) is established will the point P fall in the quadrilateral, otherwise out of it.

The second form of components is circle. As shown in Fig. 2(c), the math expressions are:

$$Ax + By + Cz + D = 0$$

$$\left|\overrightarrow{OP}\right| < r \tag{4}$$

The difference lies in the constraint condition (4), restricting the plane to a circle whose centre is O and radius is r.

A special component, that is the tread of the artillery tire, uses the third form, band region of a sphere. Our choice is based on the fact that, the tread is cambered in axial direction, but the camber isn't obvious as the tread is narrow. Spherical band region satisfies the above requests well. Here is math expression:

$$\left|\overrightarrow{OP}\right| = R \tag{5}$$

$$\left|\arccos\left(\frac{\overrightarrow{OP}\cdot\mathbf{n}}{\left|\overrightarrow{OP}\right|\left|\mathbf{n}\right|}\right)-\frac{\pi}{2}\right| < \theta \tag{6}$$

Equation (5) is a spherical surface, whose center is O and radius is R, and point P is on the surface. Equation (6) is the constraint, where vector $\mathbf{n}$ means the axial direction. It restricts the whole spherical surface to a band region with a flare angle θ. θ determines the width of the tread, as shown in Fig. 2(d).

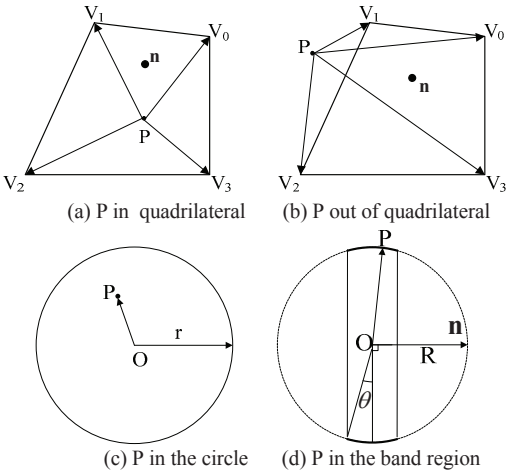


Figure 2. Three mathematical models

2) Attribute Model

The attribute model has critical information related to decontamination and could reflect results of interactions. We first introduce attributes of sensitive areas, which are part of components' attribute model.

Though sensitive areas have several types, they all possess following attributes:

a) Affiliated component: Indicating which component they belong to.

b) Type: There are four types, i.e. point, segment, arc (these three types are usually on plane surface) and curve on spherical surface.

c) Start: It's the starting point of linear sensitive areas. If the area is a point, start is itself.

d) Terminal: It's the finishing point of linear sensitive areas. If the area is a point, terminal is itself.

e) Length tolerance: Defining maximum distance the spout could deviate from sensitive area.

f) Angle tolerance: Defining maximum angle the spout could deviate from sensitive area.

g) Deviation flag: Indicating whether deviation has happened.

h) Sequence: The order in which the sensitive area should be decontaminated.

i) Actual sequence: The order in which the sensitive area is decontaminated actually.

j) Completion flag: Indicating whether the area has been washed correctly.

At component level, besides sensitive area attributes, there are other necessary attributes:

a) Form flag: Form of the component, including convex quadrilateral, circle and spherical surface.

b) Sequence: The demanded processing order.

c) Actual sequence: The actual processing order.

d) Geometry information: i.e. mathematical model.

e) Detail records: Such as the maximal and minimal value of length, angle and moving speed of the spout when decontaminating this component.

f) Completion flag: Showing whether the component has been decontaminated completely.

The attributes model will be realized by struct in next section.

IV. INTERACTION PROCESSING

On the basis of two models, we propose the processing algorithm which could handle simulation training and record changes in real-time. The algorithm mainly includes two parts: collision detection to acquire hitting point and component, and tracking sensitive areas on the component.

A. Collision Detection

There are various ways of collision detection with different accuracy and complexity. In consideration of math models in section 3, we choose ray query method. This method uses a set of lines to substitute for an object, and detect possible intersections between these lines and other primitives in 3D scene[8].

Because we have established equations for both spout and components in modeling, it's convenient to figure out intersection between the spout and some component, just combining related equations and finding the solution. If there is no solution, no intersection happens either. If there is only one solution, it's the position of the hitting point. There are no other cases.

However, there may be many intersections when components are overlapped in spout's direction. Only the nearest intersection is the actual hitting point. This can be found out easily by parameter t in (1), which reveals the distance from starting point of the spout to the hitting point.

Part of the struct designed for the attribute model of components is shown behind:

```
struct Component{
int      AssignedSequence;
int      ActualSequence;
bool     IsProcessed;
}
```

The struct member 'AssignedSequence' keeps the demanded decontaminating order. 'ActualSequence' records actual processing order in training. The boolen member, 'IsProcessed', is a flag and initialized to false. It will change to 'true' only when all sensitive areas on this component have been washed. In practice, one component may be in three states: completely not, partly or thoroughly decontaminated. These three states can be indicated by struct members 'ActualSequence' and 'IsProcessed' (Table I). When one component's been thoroughly decontaminated, it changes nothing if douched again, just skipping unnecessary processing.

TABLE I. COMPONENTS' THREE STATES

states	ActualSequence	IsProcessed
completely no	0	false
partly	non-zero	false
thoroughly	non-zero	true

B. Sensitive Areas Tracking

Section 2 has mentioned that sensitive areas must be washed during decontamination. So after acquiring collision point and component, we will further inspect whether the collision point hits areas and moves without deviation. This involves the procedure we called sensitive areas tracking. We have different tracking approaches for different kinds of areas.

1) Point: For point area, part of its struct is as this:

```
struct   PointZone{
Vector3      Start;
Vector3      End;
float        LTolerance;
int          ActualSequence;
bool         IsProcessed;
}
```

Member 'Start' and 'End' are the same point. The tracking strategy is simple: once collision point reaches the point area, i.e. the distance between them is less than length tolerance, the point area will be supposed to be decontaminated. Update the 'ActualSequence' and change 'IsProcessed' to 'true'.

2) Segment on plane: Here is the struct of linear sensitive area:

```
struct   SegmentZone{
Vector3      Start;
Vector3      End;
float        LTolerance;
bool         MissTrace;
int          ActualSequence;
bool         IsProcessed;
}
```

We can refer to Fig. 3(a) when talking about the tracking strategy. If collision point P reaches starting point A of segment AB and AB is not processed, ensure deviation flag 'MissTrace' is false and begin to track this segment. If the vertical distance from subsequent P to AB exceeds the length tolerance variable 'LTolerance', change 'MissTrace' to true and stop tracking; otherwise keep on tracking until P reaches B, and AB is decontaminated correctly. Modify the 'ActualSequence' and change 'IsProcessed' to 'true'. The tracking is done.

The vertical distance h can be calculated by AP and $\angle PAB$. Deviation will be judged by:

$$h = AP\sin\theta < LTolerance$$

3) Arc on plane: The struct of cambered sensitive area is similar to that of segment above, just adding a member 'radius' which keeps the radius value of the arc. Referring to Fig. 3(b), the tracking strategy is almost same with the above too, except for the criterion of deviation:

$$\left|OP - radius\right| < LTolerance$$

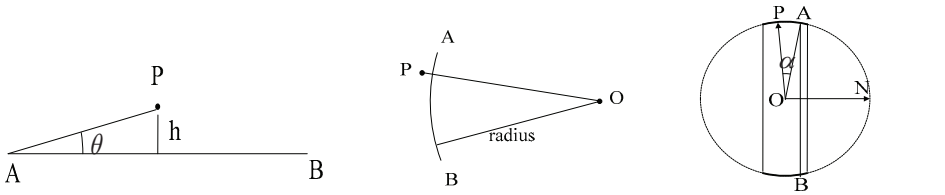
4) Curve on spherical surface: Compared to the struct of segment, two more member variables are added for curve.

```
struct   CurveZone{
( omit similar part )
float        Angle;
float        ATolerance;
}
```

Member 'Angle' shows position of the curve, and 'ATolerance' means angle tolerance. Fig. 3(c) is the front view of tread of a tire. Curve sensitive area AB is on the tread, and it belongs to a slice which is vertical to axial direction $\overrightarrow{ON}$. Thus every point of AB has the same flare angle relative to $\overrightarrow{ON}$, all equal to $\angle AON$. This value is kept in 'Angle'.

Tracking strategy is as following: when collision point P reaches starting point A of curve AB and if it's not been processed, begin to track curve AB. If angle α formed by subsequent P as in Fig. 3(c) is larger than 'ATolerance', set 'MissTrace' to 'true' and stop tracking. Otherwise continue until P reaches the end B, tracking is done completely. The criterion of deviation is:

$$\alpha = \left|\angle PON - Angle\right| < ATolerance$$



(a) Segment on plane (b) Arc on plane (c) Curve on spherical surface

Figure 3. Sensitive areas tracking

C. Processing Algorithm

As mentioned before, the processing algorithm is integrated into graphic rendering loop. So every frame it will first check whether a water spout is jetted, and then detect collisions, and finally go deep into sensitive areas tracking if necessary. When certain condition is met, some step may be skipped to increase algorithm efficiency. Fig. 4 shows the flow chart of our algorithm.

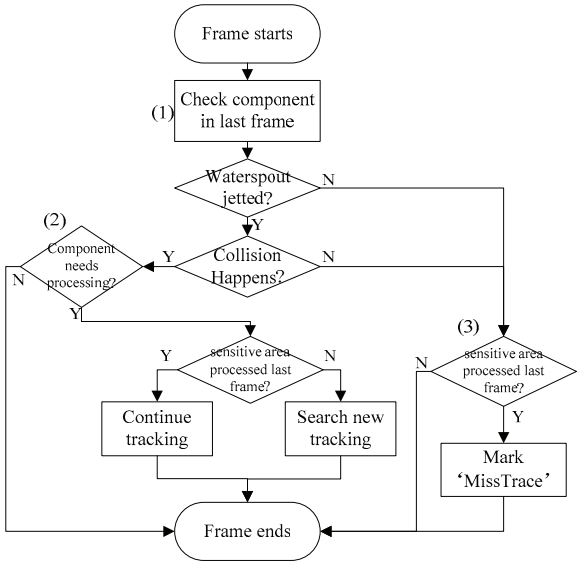


Figure 4. Flow chart of processing algorithm

There are three notes in Fig. 4:

(1) Checking component in last frame means if some component was hit in last frame, check whether it's been decontaminated completely. If so, update its state to avoid being processed again.

(2) When collision happens, if the collided component has been decontaminated thoroughly, jump to the frame end directly.

(3) If no water spout is jetted or no collision happens in current frame, but some sensitive area was tracked in last frame, algorithm will mark it by 'MissTrace' and stop tracking.

D. Evaluation

The processing algorithm would update related attributes of components in training, and these records are basis of evaluation. Assessment in reality usually allocates different weight for different index, and figure out the weighted sum as the training score.

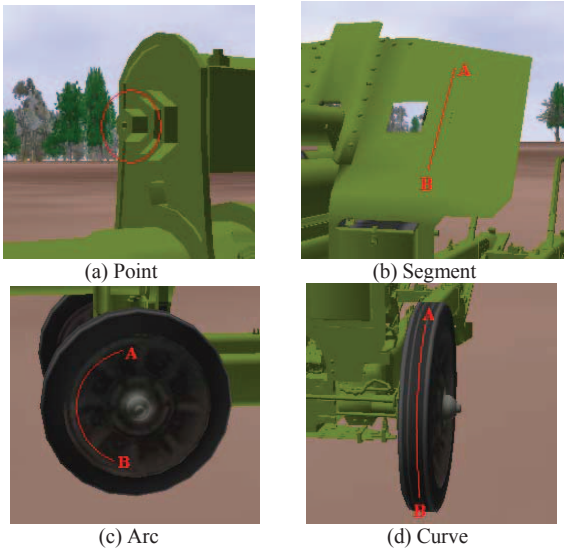
Analytic hierarchy process (AHP) [9] has the same idea with the practical assessment, but it's more theoretical and normative. According to its steps, we can obtain weight assignment reasonably. Afterwards automatic score will be done easily.

However, it's the training records which determine the training result. The evaluation method just makes it quantifiable and obvious. In fact we do save training records as text document so that trainee could gain detailed feedback. So in next section we will test and examine our algorithm by records directly.

V. TEST AND VERIFICATION

It's an effective way to test and verify the simulation via comparing outputs of both simulation and real case under the identical inputs [10]. Therefore we conduct the verification by designing tests at two levels.

A. *Decontaminate sensitive areas on every single component, and observe state changing.*



(a) Point (b) Segment

(c) Arc (d) Curve

Figure 5. Various sensitive areas on components

In Fig. 5 we use red solid lines to mark four types of sensitive areas on different component surface, yet they are invisible in fact. And when douching them, we should keep the spout hitting close to them, just as Fig. 6. Collision point P should not deviate too much from segment AB.

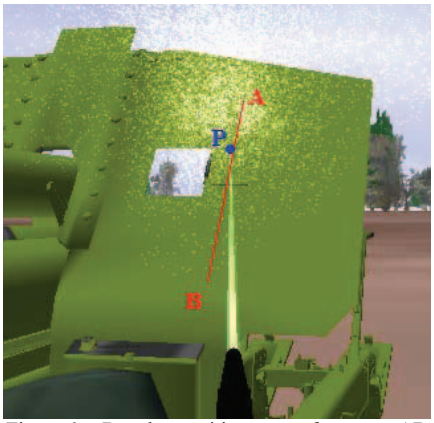


Figure 6. Douche sensitive areas of segment AB

There is no problem in tracking above four types of sensitive areas normally, i.e. beginning at the starting point, going along the areas and ending at the terminal point. So we design cases of wrong operations:

1) Not adequately approximate to the starting point but still douche further;.

2) Exactly begin from the starting point but deviate from the area before reaching the terminal;

3) Exactly begin from the starting point but stop for a while during douching and then continue;.

4) Other steps are all right but not in correct order;.

5) Not in correct order and case 2) or 3) also happens.

Just take the segment area in Fig. 6 as an example. Assume its correct sequence number is 1 and use 'WRONG' to mark incorrect order. Table II shows records of attributes.

TABLE II. RECORDS OF SENSITIVE AREAS' ATTRIBUTES IN DIFFERENT CASES

case	ActualSequence	MissTrace	IsProcessed
1	0	false	false
2	1	true	false
3	1	true	false
4	WRONG	false	true
5	WRONG	true	false

Table II has contained all false operations that may happen in reality. From the attribute records in every case we could easily conclude which mistake, sequence error, deviation or not being completely processed, has occurred. Results of douching sensitive areas will dramatically affect the scoring of the component they belong to.

B. Decontaminate multiple components and focus on state changing at component level.

Referring to Fig. 7, we mark two components by red number '1, 2' and they are meanwhile assigned sequences of those components.

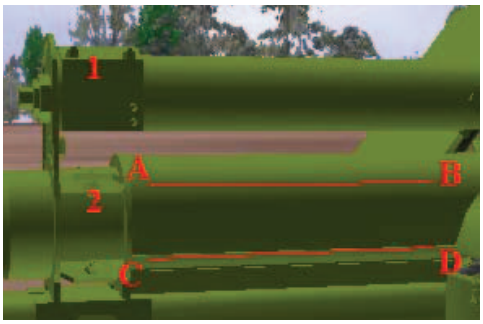


Figure 7. Tests at component level

Omit sensitive areas on component 1, and assume that there are two segment areas on component 2, segment AB and CD, marked by red solid lines. At component level, there is no doubt that only after AB and CD are processed will the completion flag, 'IsProcessed', of component 2 be set to be true, just as note (1) mentioned in Fig. 4.

So we design following cases to test arbitrary douching between two components, still using 'WRONG' to indicate incorrect order:

1) First douche component 1 partly and then turn to component 2;

2) Do as case 1), but return to component 1 to process thoroughly;

3) First douche component 1 completely, and then turn to component 2, but back to component 1 again;

4) First douche component 2 partly and then turn to component 1 to process completely;

5) Do the same as case 4) and turn to component 2 again to douche thoroughly;

6) First douche component 2 completely and then turn to component 1 to process thoroughly and back to component 2 again.

Arbitrary decontaminating more components can be regarded as combination of cases in table III and obtain similar records. Table III demonstrates if a component being douched is not what should be processed, it will be marked as 'WRONG' sequence, no matter whether it's been processed thoroughly. Otherwise it gets a right sequence number.

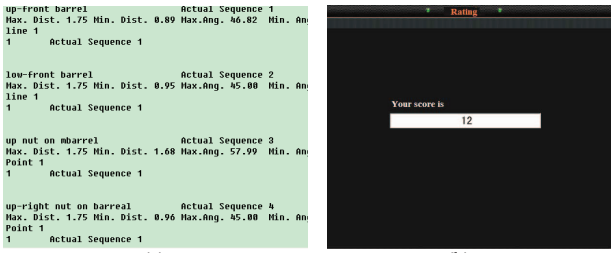
At component level, the maximum and minimum value of length, incident angle and moving speed of the water spout while douching certain component will also be kept as basis of evaluation.

Two level tests have obviously demonstrated that our simulation can indeed dispose nearly all possible interactions as in realty and take down crucial information. The information is recorded as text document, as Fig. 8 (a)

displaying, which will provide trainees with clear guidance. Based on the records automatic scoring would be done as shown in Fig. 8 (b).

TABLE III. RECORDS OF COMPONENTS'' ATTRIBUTES IN ARBITRARY DOUCHING CASES

case	Component 1		Component 2	
	Actual Sequence	*Is Processed*	*Actual Sequence*	*Is Processed*
(1)	1	false	2	false
(2)	WRONG	true	2	false
(3)	WRONG	true	2	false
(4)	1	true	WRONG	false
(5)	1	true	WRONG	true
(6)	1	true	WRONG	true



(a) (b)

Figure 8. Records and scoring interface

Not only artillery, we also apply our approach to other equipment, for example a vehicle in Fig. 9, and it works well, showing excellent generality. We realize this simulation on platform VS2005 and OGRE with AMD Athlon 1.8GHz CPU and nVidia Gefoce 7600GT GPU, and the frame rate is above 100fps.

Figure 9. Application to a vehicle

VI. CONCLUSION

We have presented a novel, comprehensive and interactive approach for training simulation of equipment decontamination. As core parts of the approach, the first two steps, modeling and processing algorithm, have achieved accurate monitoring in virtual scene, comprehensive handling of interactions, and detailed recording during simulation procedure, but still have ideal frame rate. The entire incorporate approach has demonstrated several attractive features. (1) It has improved the ability of handling interactions as in real world, which greatly augments the feeling of reality. Only the combination of realistic interactions and scene displaying will make trainees immersed in the virtual environment and gain good training effects. (2) It provides quantitative evaluation as well as detailed text feedback. Score may give trainees an intuitive understanding on their training result, but detailed feedback would better help them figure out what's their problem and where to make effort. (3) It possesses excellent generality in the simulation field of equipment decontamination. The modeling method, processing algorithm and evaluation can be applied to other equipment and still work well, so it does reduce wastes on valuable resources. In a word, our approach for training simulation of equipment decontamination has fixed previous problems and brought in comprehensive virtual training effects. It has been used in practice and can be a powerful substitute for real training.

REFERENCES

[1] Vardi A, Levin I, Berkenstadt H, Hourvitz A, Eisenkraft A, Cohen A, and Ziv A, "Simulation-based training of medical teams to manage chemical warfare casualties", Isr Med Assoc J, Israeli Medical Association, 2002, 4, pp. 540-544.

[2] William LeRoy Heinrichs, Patricia Youngblood, Phillip M. Harter and Parvati Dev, "Simulation for Team Training and Assessment: Case Studies of Online Training with Virtual Worlds", *World Journal of Surgery,* Springer New York, 2008, 32(2), pp. 161-170.

[3] SUN Xun, ZHU Xue-zheng, WANG Ping-yi, WU Yao-xin, WANG Tao, and TAO Wei-zhao, "Design and Implementation of Simulation Training System for Jet Exhaust Vehicle", *National Conference on Computer-Aided Design and Computer Graphics'06*, Jinan, 2006, pp. 436-440.

[4] ZHU Xue-zheng, SUN Xun, LIAO Ming-xin, and LI Li-sheng, "Hardware-in-the-loop Simulation Training System for Spraying Vehicle", *Computer Simulation,* The magazine agency of COMPUTER SIMULATION, 2006, 23(11), pp. 296-297.

[5] SUN Xun, ZHU Xue-zheng, WANG Ping-yi, LI Li-sheng, ZHANG Xue-feng, and YAO Wei-zhao, "Design and Implementation of Virtual Battling Simulation Training System for Decontamination Vehicle", *Journal of System Simulation,* Chinese Association for System Simulation, Beijing, 2006, 18(S1), pp. 213-214, 218.

[6] SUN Xun, ZHU Xue-zheng, MA Wei, and YU Hong-yan, "Research in Simulation Training of Jet Exhaust Vehicle Releasing Smoke Screen", *Journal of System Simulation,* Chinese Association for System Simulation, Beijing, 2007, 19(S2), pp. 518-519.

[7] PLA Academy of Military Sciences, *Military Term*, Military science press, Beijing, 1997.

[8] Tomas Akenine-Moller, and Eric Haines, *Real-Time Rendering (Second Edition),* A.K. Peters Ltd., 2002.

[9] DU Dong, PANG Qing-hua, and WU Yan, *Modern Comprehensive Evaluation Methods and Selected Cases (Second Edition),* Tsinghua University Press, Beijing, 2008.

[10] HUANG Ke-di, and ZHA Ya-bing, "A Survey on The Credibility of System Simulation", *Journal of System Simulation,* Chinese Association for System Simulation, Beijing, 1997, 9(1), pp.4-9.