Evaluating Game Technologies for Training

Dan Fu, Randy Jensen
Stottler Henke Associates, Inc.
951 Mariners Island Blvd., Suite 360
San Mateo, CA 94404
650-931-2700
{fu,jensen}@stottlerhenke.com

Elizabeth Hinkelman
Galactic Village Games, Inc.
119 Drum Hill Rd., Suite 323
Chelmsford, MA 01824
978-692-4284
elizh@galactic-village.com

Abstract—In recent years, videogame technologies have become more popular for military and government training purposes. There now exists a multitude of technology choices for training developers. Unfortunately, there is no standard set of criteria by which a given technology can be evaluated. In this paper we report on initial steps taken towards the evaluation of technology with respect to training needs. We describe the training process, characterize the space of technology solutions, review a representative sample of platforms, and introduce evaluation criteria.

TABLE OF CONTENTS

1. INTRODUCTION

The use of game-based training is rapidly gaining momentum. Games offer the benefit of experiential situated learning delivered in a dynamic and engaging manner (Macedonia, 2002). From the Army's acclaimed "America's Army" recruitment game to various medical games, a number of "serious games" have been developed for various domains.[1][2]

Despite these successes, the use of games for training presents challenges that limit its applicability. Foremost, the technology choices available today offer a bewildering array of 3-D engine toolkits, path planners, physics engines, network infrastructures, game AI, existing 3-D game source code, art asset tools, and so forth. While current efforts have yielded disparate successes, there still does not exist a unified evaluation framework by which technologies can be evaluated. Moreover, comprehensive solutions rarely exist based on a single technology platform.

[1] 1-4244-1488-1/08/$25.00 ©2008 IEEE.

[2] IEEEAC paper #1203, Version 8, Updated 2007:12:14

Given that pre-existing software can enable rapid, cost-effective game development with potential reuse of content for training applications, we discuss a first step towards structuring the space of technology platforms with respect to training goals. The point of this work isn't so much to espouse a leading brand as it is to clarify issues when considering a given piece of technology. Towards this end, we report the results of an investigation into leveraging game technologies for training. We describe the training process, outline ways of creating simulation behavior, characterize the space of technology solutions, review a representative sample of platforms, and introduce evaluation criteria.

2. TRAINING PROCESS

Since the particular strength of game technologies is to allow trainees to visualize or experience desired behavior, we focus on demonstration-based training. Demonstration-based training can be thought of as an instructional program that incorporates a sequence of five core elements: information (e.g., novel maneuvers), demonstration (e.g., demonstrations or recorded examples of task performance), practice-based methods (e.g., simulation), feedback (i.e., subsequent training based on practice results) (Salas, Priest, Wilson, & Burke, 2006).

One of the core strengths of any training strategy is that learners are guided through the acquisition of several types of knowledge and skills (Rosen, Salas, & Upshaw, 2007). While the information element focuses on declarative knowledge, the demonstration element emphasizes procedural knowledge. In the practice-based element, learners can develop procedural knowledge as well as strategic knowledge by performing a task under various conditions (i.e., different scenarios). The scenarios used for demonstration and practice-based elements capture the relevant knowledge, skills, and attitudes (KSA's) and provide examples of demonstrations as well as opportunities for the learner to engage in practice that employs the KSA's.

3. CONSTRUCTING TRAINING SIMULATIONS

In a technology context, the construction of training simulations features the behavior of simulated entities and

objects; indeed, it is what makes a simulation "come alive" for the user. Creating simulations of human behavior takes three major forms: *authoring as acting, authoring as scripting,* and *post production*.

Authoring as Acting

Human users of a simulation technology can perform a demonstration or interact with learners in the simulated world. They may perform motions which are measured by hardware and translated into character "avatar" motions in the simulated world. They may also use more traditional keyboard and mouse interfaces to drive avatars in the simulated world. The attraction of this method is that a team with experience in a particular task may be able to generate a demonstration of that task easily, or interact with learners, with a minimum of skill required to drive the simulation. This method requires planning and coordination, rehearsal, and subject matter experts (SME's). It also requires the simulation system, complete with appropriate graphics, physical models, and I/O devices as well as support personnel.

If the technology supports a full complement of SME's and trainees, it is possible to conduct live scenarios within the simulated world. These live scenarios may be conducted either as realistically as possible, or complete with instructors, narration, pauses, or pointing out of elements.

Authoring as Scripting

In this approach, the primary content of a demonstration or the "canned" behavior during practice is supplied in advance, through development of software, graphics, scenarios, and behaviors of synthetic characters. (In military parlance these characters are referred to as SAF – semi-automated forces; in commercial games they are referred to as NPC's – non-player characters.) Commercial game development studios maintain suites of tools for this purpose; ideally, scenario authors can deploy existing resources without resorting to software programming. This method requires skilled developers but leads to immersive, interactive training sessions without the need for SME's at the time of the demonstration.

Authoring as Post-Production

After behaviors have been constructed and recorded in the simulated world, it may be necessary to replay, edit, and mark up the session record to create a more informative presentation. There are two principal forms of post-production. The first relies on screen capture and video editing, leading to a digital movie whose playback can be controlled by the trainee or an instructor. The second relies on replay of the session events using a graphics rendering engine. This can be used to produce "movies" as well, but allows for more sophisticated camera control (optimized points of view and zoom-ins) or more interactive playback.

Realistically, most simulation-based training will require a combination of these approaches to authoring. The acting and scripting approaches can be thought of as opposites on a spectrum of authoring complexity, while post-production is a necessary step to transform recorded content to useful demonstration or feedback.

4. REVIEW OF TECHNOLOGIES

In terms of virtual environments, there is a great variety of platforms with varying strengths and weaknesses, largely by nature of the differences in the domains they respectively simulate. This portion of the research effort sought to review a representative set of platforms at the category level. The resulting technology review targets performance criteria pertaining to specific authoring capabilities and use cases, as opposed to a general survey. In this section we summarize potential technology platforms, establish evaluation criteria, and describe specific platforms.

Game Engine Types

There are several commercial game technologies upon which to base authoring. With the growth of the videogame industry, game development platforms have emerged that offer a powerful array of authoring capabilities. Broadly, platforms can be characterized according to two categories: depiction and plurality. The most popular type of depiction is 3-D, which tries to make the visualization as realistic as possible. The other type is 2-D which is not as realistic. Plurality is the number of players that can participate: single player, multiplayer, or massively multiplayer online (MMO). Using these two dimensions, Table 1 shows the range of platform technologies.

With an eye towards authoring, the multiplayer and massively multiplayer platforms encourage an "authoring as acting" approach. Individuals involved in the creation of a demonstration can participate by playing their individual team roles. This approach is less viable for single player platforms as coordination of several individuals, acted by a single author, is difficult. Thus, the "authoring as scripting" method has the author specify the behavior of SAF/NPC in a demonstration. In the following subsections we describe each of the basic technology platform categories.

2-D Games—Two dimensional displays may provide great ease of authoring for certain types of tasks. 2-D game engines depict a point of view either from overhead or from the side. For example, a videogame such as Pac-Man shows an overhead view. The heights of the notional halls that are traversed are not depicted. A videogame like Space Invaders shows a view from the side. These types of game engines are early technology and are less demanding than 3-D engines both in terms of memory and computation. Because they lose one dimension, they use symbols ("sprites") that represent actors or objects in the videogame.

A game that features "scrolling" has the added capability to give the player a sense of motion across a terrain. Games like Pac-Man and Space Invaders are static in that the point of view of the player never changes. In contrast, a game like Gauntlet or Defender has the player's avatar always near the center of the screen. As the avatar moves across terrain, the point of view stays with the avatar. An example of a modern PC game relying on 2-D graphics is Civilization II. Its production values are far higher than the '80's games.

2-D depictions could be most promising for "big picture" explanations. For example, the coordinated movement of a fire team requires that all members know their roles, the actions expected of them, and of each other for cross training (Salas & Cannon-Bowers, 2000). Many games played primarily in 3-D retain a 2-D view to provide context and situational awareness. This is especially true for understanding coordinated movement. Some examples include GuildWars and RuneScape: massively multiplayer online role playing games, and Battle for Middle Earth: a PC strategy game. Figure 1 shows a screenshot of a Clemson University visualization tool for recorded movements of Marine fire teams in Military Operations in Urban Terrain (MOUT) scenarios (MOUT, 2006). The view shows a 2-D overhead view of locations of soldiers and their direction of sight. On the upper right is a camera recording.

2-D game engines offer the lowest amount of fidelity as they are almost incapable of rendering a realistic scene. Thus, what the player sees is essentially a diagram. Oftentimes 3-D games will include a 2-D overhead display to provide useful information that would otherwise be difficult to see in a (crowded) 3-D display.

3-D Single Player Games—A 3-D single player game features a 3-D graphics engine that displays the game environment by rendering it from parameters and descriptions of 3-D objects. The engine renders the environment from a 3-D scene. It assumes the player is the only person operating in the environment, and that anything else moving (whether supposedly human or not) is controlled by the engine. The engine may support many "cameras" or viewpoints within the environment, such as first person, tethered, overhead, or a user-controllable point of view. It may support display of several cameras simultaneously on one screen. A game such as Half-Life is a good example where most of the simulation is constructive, including several types of opposing entities. There are three major genres of single player games:

First person shooter (FPS): Depicts the 3-D scene from the point of view of the player who is usually armed with a gun. The emphasis is on high tempo tactical motion requiring "twitch" skills which require the player to exercise fine control to (typically) shoot at monsters.

Real-time strategy (RTS): These videogames depict 3-D scenes, but from an "in the air" perspective, typically 45 degrees to the terrain's plane. The player controls a team of avatars against an enemy force. Time is spent creating forces, attacking enemy forces, and exploring the world. The AI for forces isn't complex. During battle is when the player's ability to task forces becomes crucial as the player must constantly monitor outcomes of conflicts and re-task and re-group units across a terrain.

Role-playing game (RPG): These videogames are "dungeons and dragons" simulations that have the player assume control of a small team typically less than a dozen characters. The point of view is similar to RTS, but without time pressure. The player is given the ability to pause the simulation to decide what to do. Indeed, the menu of possible actions for each character is quite rich, thus the player may create a rich choreography of actions for each.

These three types of videogame genres are broad. Several games mix two types. For example, Metal Gear Solid has the player controlling a single character with an RTS point of view with a semi-transparent overhead 2-D view in the upper right. As well, when desired the player can switch to a FPS point of view in real time for the purposes of observing or shooting. CHI's VECTOR (McCollum et al., 2004) has a FPS point of view, but no twitch factor: it gives the player time to decide what to do or say next (similar to an RPG). Stottler Henke's Informant (Cramer, Ramachandran, & Viera, 2004) has a FPS point of view as well, but NPC's vary reactions depending on when the student says an utterance.

The use of 3-D graphics (along with many other technological advances since the '80's) allows more realism than 2-D. FPS games are the most realistic as they attempt to depict the scene in the most life-like manner possible. RTS and RPG are in between 2-D and 3-D. Indeed, early RTS games could be considered scrolling 2-D games. Even though character models are generated in 3-D, their subsequent depiction and animations always appear the same way as the player's point of view is always at a constant angle via orthographic projection.

The industry trend of recent years is toward hyper-realism of graphics. Water must be translucent; individual hairs and blades of grass must wave in the wind. Game developers have identified a phenomenon known as "the uncanny valley", where rendered images of humans fail to display matching realism in facial expressions and gestures. The resulting displays are viscerally "creepy" to viewers, to the point where they prefer less realistic images instead. Today's 3-D engines such as Half-Life 2 (Keighley, 2004) implement facial expressions based on studies done by Paul Ekman (Ekman, 1974).

Table 1: Basic Categories of Game Engines

		DEPICTION	
		2-D	**3-D**
PLURALITY	**Single**	A player controls an avatar in a 2-D environment. The perspective is an overhead or side view.	The player's avatar operates in a 3-D environment. First-person shooters and real-time strategy games are common.
	Multiplayer	Multiple players control an avatar in the 2-D world. Typically an extension of single player. Much less common.	Typically less constructive than the 3-D single player games as there are human players.
	Massively Multiplayer	A small handful exist as free games. Several commercial 3-D based systems have ancillary 2-D views as well.	Similar to 3-D multiplayer except players can number in the thousands with persistent worlds.

Figure 1: Visualization tool showing recorded fire team activity.

3-D Multiplayer—A multiplayer game engine is less constructive than the 3-D single player in favor of increasing the number of human players involved (usually ranges from 2 to 64). One might think of multiplayer as the same as single player except that the AI for avatars is supplanted by real human control. There are two implications for this style of gameplay:

Importance of social contact: Players are competing or cooperating with each other. The emphasis is on human-to-human gameplay. There is always a way for players to communicate during the game, either for teamwork or social purposes.

Absence of storyline: The entertainment that players derive is from the satisfaction of competing against each other. Contrast this to developing storylines in single player games. A game such as Half-Life has what might be considered an intricate story as opposed to games like Quake 2 which barely have a plot. Multiplayer games have themes related to single player, but time is reset every few minutes and the game starts over. Terrain requirements are typically minimal.

Almost all modern 3-D multiplayer game engines also feature single player modes; in fact, the earliest games such as Quake and Half-Life started as single player games that later featured multiplayer online "death match" games where players would fight and accumulate points based on how many times they eliminated opponents. Later, games such as Counter-Strike emerged which featured team-based gameplay. The addition of headsets with microphones enabled players to communicate as if on a radio net.

3-D Massively Multiplayer—One might think of MMOG's (massively multiplayer online games) as similar to 3-D multiplayer except on a bigger scale. They feature a huge virtual world where one may explore and meet other avatars controlled by other players. There aren't really so much levels or scenarios as there is a single ongoing world simulation. The scale and corresponding technical requirements (high capacity internet infrastructure, persistent worlds) on these games differentiate it from a 3-D multiplayer game.

The social component of MMOG's is much more important than in multiplayer as the virtual worlds are persistent. Unlike the multiplayer games whose participants can

change minute by minute, round to round, or server to server, MMOG players build experience and history in the virtual world. The world changes and so do the avatars in it. For example, in Everquest players can attain certain weapons or experience points that imbue them with augmented power in the world. There is, however, a trend towards "instancing," where a particular part of the world will be replicated for each party wanting to enter such as GuildWars.

5. EVALUATION CRITERIA

Our investigation reviewed self-contained virtual environments, or mixes of technologies that could conceivably serve for authoring providing 3-D views of the virtual world. This included SAF-centered options such as OneSAF coupled with a visualization tool or Half-Life DIS. Multiplayer game engines included Renderware, Unreal, Gamebryo, and Jupiter, as well as specific applications such as VBS1 (Virtual Battlespace 1) which is based on Operation Flashpoint. Massively multiplayer online games (MMOGs) included OLIVE (Online Interactive Virtual Environment), BigWorld, and Second Life.

Considering the types of technologies available, we partition them into a "platform space" with simulation and videogame technologies as a focus. **Figure** 2 shows the space of technologies we have surveyed. Each space is marked with a letter. Broadly speaking, most platforms for the military fall into sections c, e, and f.

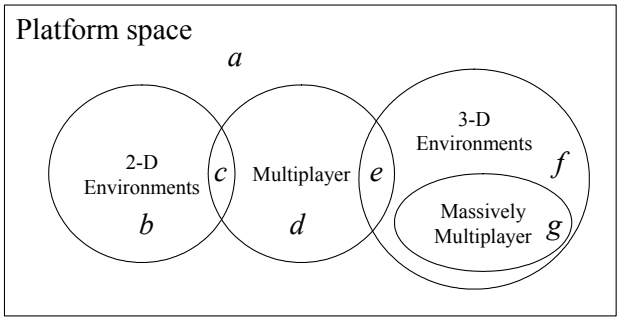

Figure 2: Venn diagram of platform technologies.

We initially defined a thorough list of evaluation criteria for the virtual environments that could be considered for authoring. From that list, no single platform meets all criteria, so we attempted to narrow the list to a central set of criteria for our assessment. The following list shows our criteria with an emphasis on deployment into DoD. Table 2 provides examples for each space for simulation and games.

Native scenario authoring capabilities. Authoring of scenarios can be very difficult. This is a major concern for game-based training, and therefore the availability of tools for authoring is an important evaluation criterion. Where new art assets or animations or models or terrain are required for a scenario, the upfront pre-production tasks are significant for all platforms. However, if these tasks are held as a constant and we consider the next scenario-specific pre-production stage which involves assembling the elements of a scenario together on the terrain, some platforms make this step nearly trivial, while others require specific tools expertise or even programming expertise. Clearly the advantage is with the former.

Native support for synchronized communications. There are training domains in which communications are not a major component. However, we believe those domains involving communications are more deserving of focus for an authoring tool. Simply by the nature of team operations, and many contemporary asymmetric warfare challenges in which coordinated information is both dynamic and mission-critical, the depiction of synchronized communications is likely to be a key element of the training goals for a wide variety of instructional demonstrations.

Interoperability. We considered platform inter-operability in the sense of 1) access to the code either through direct source code availability or an SDK/API, 2) compatibility with existing standards for simulation event data such as DIS or HLA as a means for potentially including SAF behaviors from military simulations such as OneSAF, and 3) the absence of programmatic roadblocks to constructing an integrated solution.

Capture. Once a demonstration has been constructed, the ability to transmit it becomes an important resource multiplier. We anticipate the dissemination of demonstrations and feedback delivery to be a critical factor. It is ideal for a platform to ultimately support seamless export mechanisms in tools, ideally with an existing or developable native video capture capability. A component of this criterion is also the requirement that the engine architecture fundamentally supports logging and playback of execution events.

Modeled subject matter. The availability of existing art, animations, and models for objects appearing in military training domains is a significant positive factor. While these elements can be produced when absent, there is clearly reason to prefer platforms that have such assets within easy grasp already. Additionally, a key element of this criterion is specifically the capability of a given virtual environment to model combined platforms with mounted and dismounted views and maneuverable vehicles.

Additionally, several of our early criteria considered as performance factors for various platforms were determined to have relatively less importance. For example, the availability of automated entity behaviors may have only limited value for demonstrations and practice, particularly when team tasks are being depicted. A common reported finding is that semi-automated force behaviors are useful for some limited tasks like crowd "filler" kinds of behavior, but

the key actions are more complex than what SAF can enact. Therefore for time-intensive authoring, at least for demonstration purposes, an approach involving human role-players in a multiplayer or massively multiplayer platform appears to be more practical, even with the customary difficulties associated with coordinating human role-players.

Table 2: Examples of Platform Space.

Space	Simulation	Game
a	All simulations including operations research	All games
b	Wargames, Course of action analysis	Solitaire, Space Invaders, Gauntlet
c	Aide de Camp 2, SimVentive, play-by-email wargames	Abuse, older RTS's
d	Board wargames	Multiplayer card and board games
e	DIVAARS, Game DIS (Half-Life 2), VBS1 / Ambush! (Operation Flashpoint), Microsoft Flight Simulator	Doom 3, Counter-Strike, Quake III Arena, Starcraft
f	Tactical Iraqi (Unreal Engine), VECTOR (Jupiter), Informant (Jupiter)	Half-Life, Gran Turismo, StarCraft
g	OLIVE, Second Life	BigWorld, Everquest, World of Warcraft

6. PLATFORM EXAMPLES

In this section we describe some representative examples in the constellation of technologies that could be leveraged.

Torque Game Builder, 2-D

Torque Game Builder is a comprehensive toolkit for building 2-D games. It is composed of a UI builder, custom scripting language, 2-D game engine, physics engine, and a networking package. The tool software runs on a Windows PC, but the resulting games can also be played on Apple OS X and Xbox 360. Documentation for the product is extensive with online manuals and a developer's forum. Sample games are included.

Unreal Engine, 3-D Multiplayer

Unreal Engine is a popular game engine marketed as such in the multiplayer game category. Unreal Engine 2 and its Xbox variant are used in many current games, including most versions of America's Army. Here we concentrate on the recently released Unreal Engine 3.

Unreal Engine 3 runs on 2006 and later hardware, requiring DirectX 9 or next generation consoles. It has peer-to-peer networking support for up to 16 players, and client-server Internet support for up to 64 players. The network supports mixing of PC and console devices in a single game session. The framework provides numerous third party components for such functions as AI, vehicle physics, and facial expressions.

One advantage of Unreal Engine 3 is the ability to create state of the art graphics for several platforms simultaneously (though not optimized for any). Another is the extensive toolset which features tools for particle effects, 3-D audio effects, collision effects, organizing animations and meshes, visual scripting, materials editor, user interface, and content organization. Many of these tools are available to the public to allow players to create their own game modifications. A significant disadvantage is cost.

Unreal Engine 3 is used for Gears of War (FPS) and Unreal Tournament 3 (Multiplayer FPS), as well as America's Army 3 and numerous games in production by major studios. Unreal Tournament 3 will be distributed with authoring tools for both graphics (Unreal Editor) and behaviors (Kismet), for player mods.

Gamebryo Element, 3-D Game Engine and Tools

Gamebryo Element is a 3-D graphics engine that features platform optimizations for PC and next-generation consoles. Its particular strength is its flexibility. Rendering capabilities include fully customizable rendering pipeline, vertex and pixel shaders, shadows, bump maps, and screen-space geometric primitives. Gamebryo can integrate with physics engines, but does not feature its own.

COTS tools include 3-D Studio Max and Maya graphics plug-ins, a suite of animation and of runtime performance tools, and Scene Designer, for integrating art assets, light sources, and cameras and verifying the rendered output to create scenes, levels, and worlds.

Titles produced with Gamebryo Element include Civilization IV and numerous Elder Scrolls titles. Military applications include Cubic Defense Systems' multi-player EST 2000 Small Arms Trainer, and BreakAway Games' 24 Blue, a Navy flight deck operations simulator.

Machinima, 3-D Multiplayer

Machinima refers to the use of 3-D videogame technology for producing movies. What differentiates machinima from animations or cartoons is that 3-D multiplayer game engines render scenes in real-time. That is, the avatar "actors" are under control by human controllers. They interact in the 3-D world, and a virtual camera records the action. This type of technology is "authoring as acting."

Machinimation 2.0 is an example of a machinima software package based on id Software's Doom 3 game engine technology. Movie-making capabilities are supported through the use of additional control panels arrayed around the 3-D game world window.

Machinimation 2.0 affords the following capabilities:

- Avatar actions – The author can take control of an avatar and record actions. The author can also see other previously-recorded avatars as well during the recording.
- Timeline - The author can move back and forth on a timeline. Recorded avatars and camera will move accordingly.
- Overlays - Insert text or images.
- Camera motion - The author can record camera location and posture in 3-D space. Using the timeline, the author can specify where the camera should be, when. The package figures out smooth motion. Cameras can also be "chase cams" that follow an avatar at recording time. Unlike DIVAARS (Clark et al, 2004) the author cannot simply add them in later if desired.
- Game is live - All aspects of the game work during recording; e.g., monsters will attack, all sound effects will be heard.
- Preview - Play back all the recordings immediately after laying down a "track."
- Render movie - Mix everything down into a high resolution video.

Figure 3 shows a screenshot of Machinimation 2.0's authoring interface. The panel on the left invokes basic authoring actions such as inserting a new camera. The motion of the camera (if any) is articulated by using regular player controls except that the camera is not bound by gravity. Thus, the author "flies through" the game world. What the author sees is what the camera will capture. Special keys are used to record camera points. The orange lines with white points are camera paths. The points specify camera location and posture. The timeline at the bottom enables the author to select a time and see where the avatars and cameras will be. The Figure shows the recorded avatars at exactly 4.52 seconds from the start of the scene.

Although this approach to authoring is viable, the software is no longer supported. While it represents a powerful authoring method, it is excluded from our evaluation.

VBS1, 3-D Multiplayer

VBS1 (Virtual Battlespace Systems 1) is a 3-D training system for small unit tactics. It is based on the game Operation Flashpoint which was originally released in 2001 by Bohemia Interactive. It features realistic terrain, mission editor, and functional vehicles and equipment. The US Marine Corps funded the VBS1 training system which additionally includes an AAR component. Follow on funding by the Australian Defense Force resulted in the "VBS1 Instructor Interface" which enables instructors to "fly through" the mission to emplace items such as IEDs, and also monitor in-game activity (Morrison et al., 2005). Its use has slowed as of March 2006 because its successor—the MVTC (Mobile Virtual Training Capability)—was released. MVTC is a "turnkey" system which also includes all necessary hardware as well as software. BBN used Operation Flashpoint to develop the "Ambush!" convoy training system, funded by the DARPA DARWARS program.

OLIVE, 3-D MMOG

The OLIVE (On-Line Interactive Virtual Environment) platform is based on an MMO game engine, and provides a server-driven persistent virtual world (Mayo, Singer, & Kusumoto, 2005). The typical process of scenario authoring takes a simpler meaning in OLIVE as compared to the typical military simulation world. In an authoring mode, avatars can simply call up and place any objects in the virtual space. Setting up a scenario like a checkpoint operation is as simple as choosing the location, retrieving barriers and other checkpoint objects or vehicles and placing them at that location, and then having players log in, and SAF auto-generate. Individual avatars can define their appearance through a set of templates, and a variety of culturally typical templates already exist for areas of likely military operations. OLIVE is standards compatible, and supports SAF control of entities through DIS; for example OTB has been used to control crowd characters in exercises. OLIVE provides a voice over IP capability for communications between virtual avatars, and also has an existing (but simple) playback mechanism used for AAR.

So far OLIVE's use within the military has been confined to research efforts. The Asymmetric Warfare-Virtual Training Technology (AW-VTT) effort for RDECOM is intended for joint, interagency, and multinational operations in the Global War on Terror, including asymmetric and unconventional warfare, antiterrorism, force-protection and missions-other-than-war. For the AW-VTT, a notional one

square kilometer urban setting geo-referenced to Baghdad has been modeled in a 3-D virtual environment.

BigWorld (3-D MMOG)

Based in Australia, BigWorld is the leading MMO engine marketed to game developers. The BigWorld server is capable of supporting millions of users and millions of shards, with automatic load balancing and error recovery. It supports multiple games per server cluster. Within a game it supports public, restricted, and private areas (such as for quests). It supports data and behavior updates at runtime. It has configuration and load balancing tools.

The BigWorld client is scalable for both high-end and casual graphics. For authoring, BigWorld does not provide game AI but rather an API for Python-based object behavior scripting. The graphics authoring tools include a world editor, model editor, and particle effects editor, and ability to integrate with the standard graphics tools 3-D Studio Max and Maya.

The principal disadvantages of this MMO platform are cost, and the lack of any US military specific standards, behavior or graphics pipeline, graphics, or existing scenarios.

Video Capture and Conversion Tools

The ability to play back a demonstration either via video recording or through the simulation engine is a valuable capability. As a minimum, the output must be in a format compatible with a current desktop PC. Aside from machinima approaches which focus on demonstration, few platforms feature this capability. Example platforms with custom AAR include Game DIS, which is based on the Half-Life 2 engine, and OLIVE which is compatible with ARI's DIVAARS.

In principle, generation of media for demonstration and feedback is not a difficult technical challenge. COTS tools already exist for certain types of screen capture, such as TechSmith's SnagIt software which can record arbitrary screen content. Subsequent mix-down capabilities can be afforded by TechSmith's Camtasia Studio 2 which features movie-making capabilities along with advanced features such as zoom control and call outs. Open source applications such as VirtualDubMod enable the user to package content into a variety of media formats tailored for different media player hardware and software.

7. PLATFORM EVALUATION

With respect to our criteria,

Table 3 shows how our samples from the previous subsection stack up. In some cases, it was unclear whether certain assets could be available. For example, America's Army is based on the Unreal Engine. While there is certainly modeled subject matter in existence, it's unclear whether they can be easily secured for demonstration construction. In these instances, we erred on the side of optimism.

2-D game engines such as Torque Game Builder are unlikely to be used in team training systems aside for diagrammatic overhead views of team movement. Building this capability presents very few risks. The 3-D multiplayer games, such as VBS1, are an attractive set as they are all now able to support multiplayer participation. This supports the "authoring as acting" approach. MMO's also support this as well. Production platforms, such as Camtasia Studio 2, provide a way for authors to assemble a demonstration.

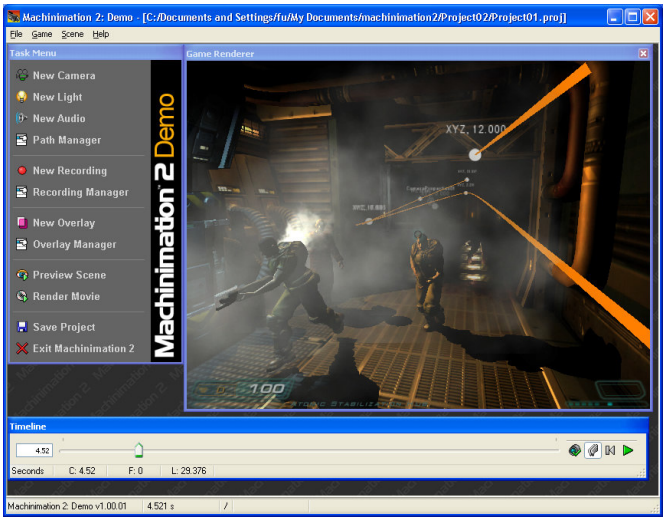

Figure 3: Machinimation 2.0 screenshot

8. CONCLUSIONS

Starting from a description of the training process, we identified where the choice of a technology platform plays a major role; namely, in the demonstration, practice, and feedback elements. We then introduced basic characteristics of game engines such as the method of depiction and number of human participants. Although taken in entirety there are hundreds of other factors to consider, as well as exotic technology features such as crowd behavior, we believe our dimensions are a good starting point for a taxonomy of platform types. These types were then subsequently situated in a "platform space" which enabled the creation of a table which provided relevant data point examples of simulations or games.

Table 3: Platform Evaluation Using Guidelines

Evaluation Criterion	2-D	3-D						Post Production
	Torque 2-D	Multiplayer				MMO		Camtasia Studio 2
		Unreal Engine	Gamebryo	Machinimation	VBS1	Olive	BigWorld	
Native scenario authoring capabilities	Yes	Yes	Yes	Yes	Yes	Yes	Yes	N/A
Native support for synchronized communications	No	Yes	Yes	Yes	Yes	Yes	Yes	N/A
Interoperability	No	No	No	No	Yes	Yes	No	N/A
Capture	No	No	No	Yes	Yes	Yes	No	Yes
Modeled subject matter	No	Yes	No	No	Yes	Yes	No	N/A
Transition paths	No	Yes	No	No	Yes	Yes	No	N/A

We introduced six evaluation criteria for adoption of a given game technology; namely, authoring capabilities, support for comms, standards interoperability, capture for demonstration or feedback, and existing subject matter models. We picked a representative sample across 2-D and 3-D multiplayer and massively multiplayer games and engines as well as post-production applications. The results of the evaluation were presented in a table.

We do not prescribe the "best" solution. In reality there has been no emergence of standard platforms for common military needs within the greater training community. COTS art assets are not commonly shared unless they are wedded to a particular platform. Most deployed training systems do not share a common base of content as standards vary. For example, MMOG's simulate a world which resides on a sphere, such as the earth. However, multiplayer games assume a flat earth. This discrepancy becomes apparent (say) during a playback of an OLIVE log file using DIVAARS. Software transformation steps are necessary to reconcile differences in assumptions.

We view this work as a clarifying step for evaluating platforms for training needs. Convergence of standard evaluation criteria for "serious games" training may eventually yield a common baseline for technologies which would ultimately benefit tomorrow's warfighter.

ACKNOWLEDGEMENTS

Eduardo Salas, Michael A. Rosen, and Christin L. Upshaw provided the training process exposition. Portions of this work were funded and sponsored by Don Lampton of the Army Research Institute under contract #W91WAW-07-P-0020. Opinions expressed are those of the authors and do not necessarily represent an official position of the Department of the Army or the Army Research Institute.

REFERENCES

Clark, B. R., Lampton, D. R., Martin, G. A., & Bliss, J. P. (2004). Virtual After Action Review Systems (DIVAARS) (ARI Research Product 2004-03). Arlington, VA: US Army Research Institute for the Behavioral & Social Sciences.

Cramer, M., Ramachandran, S., Viera, J. (2004) Using computer games to train information warfare teams. Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC).

Ekman, P. & Friesen, W.V. (1974) Detecting Deception From Body or Face, Journal of Personality and Social Psychology, 29: 288-98.

Keighley, G. (2004). The Final Hours of Half-Life 2. Retrieved December 12, 2007 from http://www.gamespot.com/features/6112889/p-5.html

Macedonia, M. Games Soldiers Play. IEEE Spectrum, Vol. 39, Issue 3, pp. 32-37, Mar 2002.

Mayo, M., Singer, M. J., & Kusumoto, L. (2005). Massively Multi-Player (MMP) Environments for Asymmetric Warfare. Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC), Paper No. 2149.

McCollum, C., Deaton, J., Barba, C., Santarelli, T., Singer, M., & Kerr, B. (2004). Developing an immersive, cultural training system. Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC).

Morrison, P., Barlow, M., Bethel, G., & Clothier, S. (2005). Proficient Soldier to Skilled Gamer: Training for COTS Success. Proceedings of SimTecT 2005.

MOUT project, (2006). Retrieved June 21, 2007, from http://www.parl.clemson.edu/~ahoover/MOUT/

Rosen, M. A., Salas, E., & Upshaw, C. L. (2007). Understanding Demonstration-based Training: A

Conceptual Framework, Some Principles and Guidelines. *Unpublished manuscript*.

Salas, E. & Cannon-Bowers, J. A. (2000). The Anatomy of Team Training. In: Tobias & Fletcher (Eds.) Training and retraining. Woodbridge, CT: Macmillan Reference USA: 312–34.

Salas, E., Priest, H. A., Wilson, K. A., & Burke, C. S. (2006). Scenario-based training: Improving military mission performance and adaptability. In C. A. C. A.B. Adler, and T.W. Britt (Eds.), *Military life: The psychology of serving in peace and combat* (Vol. 2: Operational Stress, pp. 32-53). Westport, CT: Praeger Security International.

BIOGRAPHIES

Dan Fu is a group manager at Stottler Henke Associates. He joined nine years ago and has worked on several artificial intelligence (AI) systems including AI authoring tools, wargaming toolsets, immersive training systems, and AI for simulations. Dr. Fu is the principal investigator for SimBionic, which enables users to graphically author entity behavior for a computer simulation or game. Dr. Fu holds a B.S. from Cornell University and a Ph.D. from the University of Chicago, both in computer science.

Randy Jensen is a group manager at Stottler Henke Associates, Inc., working in training systems since 1993. He has developed numerous Intelligent Tutoring Systems for Stottler Henke, as well as authoring tools, simulation controls, after action review tools, and assessment logic routines. He is currently leading projects to develop automated after action review for Marine Corps combined arms training, a framework for ITS interoperability with distributed learning architectures for the Joint ADL Co-Lab, and an authoring tool for virtual training demonstrations for the Army. He holds a B.S. with honors in symbolic systems from Stanford University.

Elizabeth Hinkelman is VP of Development at Galactic Village Games, LLC. Prior to joining Galactic Village, Elizabeth was project manager for the suite of voice Web infrastructure products at iConverse, and contributed to several other commercial applications of language processing technology. She has engaged in academic research, leading research teams and winning government funding for investigations of human and computer processing of natural language. Elizabeth received her Ph.D. in Computer Science from the University of Rochester, and a postdoctoral fellowship from the University of Chicago.