

## CHAPTER 19

# FROM VIZ-SIM TO VR TO GAMES

## How We Built a Hit Game-Based Simulation

**MICHAEL ZYDA, ALEX MAYBERRY, JESSE MCCREE,  
AND MARGARET DAVIS**

### ABSTRACT

Program managers want games for their next training simulator or combat-modeling system. Corporations want their messaging put forward in game form. These desires are sharpened by the enormously successful career of the *America's Army* game, the first "serious" large-scale game ever produced. In this chapter, we discuss why people want their next-generation simulation to look like a game and where they got that idea. We then describe the development of *America's Army* to elucidate what is required for such an effort. *America's Army's* can be studied as an example of the challenges we will encounter as we go forward with game-based simulation for training and combat modeling.



PLATE 1. Soldiers Spill from a Stryker in the *America's Army* Online Game



## INTRODUCTION

Why do so many people want games for their next training simulator? For one thing, games boast intuitive interfaces, which is one reason kids the world over spend hours playing games. The average *America's Army* fan spends something like sixty hours in the game, counting those who completed the basic-combat training, and it is only one of the top-five online games: their cumulative hours must be staggering. Ask any parent of an avid online gamer—the number of kids hooked and time spent is scandalous. Games and their interfaces have become second nature to youth.

As new games appear, they are adapted to instantly. Game interfaces are as standardized as automobile dashboards—drive one, drive them all—and in any case, setup functions allow for preferences. Because next-to-no training time is needed to joyride the latest game, attention is riveted to the story and challenges to be traversed.

Games are also attractive for their immersive qualities. As a rule of thumb, there is more immersion in a typical game than in a typical training simulator. Teenagers often enter a game world before dinnertime, after which it is difficult to prise them out to eat: need more be said? The same is rarely true of training simulators. If the training world were to achieve this level of immersion, they would have to invest heavily, as the game world does, in story and design. Training developers spend little on story and even less on design; most time and money goes to technology. Conversely, technology gets perfunctory treatment from game makers, who use entertainment tricks to convey story rather than worry about the real modeling of the displayed system.

So there are strong reasons to move our training simulations to a game basis. But there are problems.

One of the larger problems is the generation gap. Games mean “frivolous wastes of time” to the older generation, so it is hard to convince them to buy off on such training systems or even the term “game-based simulation.” Eventually this resistance will fade, but at present it is our biggest impediment. Meanwhile, we know we have to move on. When we hear stories about nine-month learning curves for the latest combat-modeling system, we can't but think of the five minutes it takes to drive a game. As a community, we want our systems to offer training in five minutes. We want our systems as immersive as games. We want them entertaining, so that work is play and people don't leave. In short, we want our training systems so immersive that soldiers forget to eat.

### Where Did We Start?

If we go back to the mid-1980s, when we launched the field now known as virtual reality, the motivation was to make 3D virtual environments available to everyone who could afford a workstation.

At that time, all we had were very expensive (multi-million-dollar) visual-simulation systems. In the NPSNET project (Macedonia, 1994; Singhal & Zyda, 1999), we deemed ourselves successful when we had over a hundred organizations

ask for tapes of the NPSNET source. We simplified lives by giving away the code, enabled anyone with a \$60K workstation to extend that code for their own purposes.

So how do we get back to such a no-frills state? Games are mainstream entertainment—and old-style virtual worlds and visual-simulation games, we harness the creativity of our users' acumen, to get our training simulators

### Why Did We Start Thinking About

The 1997 National Research Council report “Linking Entertainment and Defense” identified interactive entertainment—not digital games—the main drivers for networked entertainment developments in modeling and simulation. To examine networked entertainment for training, we thought a lot about this insight when we conducted research in modeling, virtual environments, and simulation became a focus.



PLATE 2. A Black Hawk helicopter as modeled in a virtual environment.

<sup>1</sup> The Modeling, Virtual Environments & Simulation Laboratory, Monterey, California.



ask for tapes of the NPSNET source so they could adapt it to their training needs. We simplified lives by giving away the source codes to NPSNET I through IV. We enabled anyone with a \$60K workstation to play in SIMNET and DIS simulations or extend that code for their own purposes.

So how do we get back to such a notion for games? Again, remember that games are mainstream entertainment—and big money. Games look way better than the old-style virtual worlds and visual-simulation systems we used to build. With games, we harness the creativity of artists and designers, rather than engineering acumen, to get our training simulators built.

### Why Did We Start Thinking About Games?

The 1997 National Research Council report entitled “Modeling and Simulation – Linking Entertainment and Defense” (Zyda & Sheehan, 1997) states that games and interactive entertainment—not defense research expenditures—have become the main drivers for networked virtual environments. To keep up with developments in modeling and simulation, that report indicated, DoD ought to examine networked entertainment for ideas, technologies, and capabilities. We thought a lot about this insight when forming the MOVES Institute<sup>1</sup> as a center for research in modeling, virtual environments and simulation, and game-based simulation became a focus.



PLATE 2. A Black Hawk helicopter as modeled in *America's Army*

<sup>1</sup> The Modeling, Virtual Environments & Simulation Institute at the Naval Postgraduate School, Monterey, California.



### What Does Game Development Cost?

So if we make games, what's the bill? In Table 1, we see a notional cost for *America's Army*. *America's Army* was built as an entertaining vehicle for strategic communication (Davis, 2003; YerbaBuena, 2004; Zyda, 2003a&b). We start by discussing a notional/approximate cost for that development. With luck, our training simulator will be less expensive.

The first row lists notional game-engine costs. Game engines licensing for use in one game runs from \$300K to \$1.5M. ("Game engine," by the way, is a poor term. It ought to be "game engine and authoring-tool set," as that is what you expect with your license.)

We want to get our game out in twenty-four months, so for the moment let's banish the notion of developing our own engine and toolset. Let's assume the lowest cost, \$300K, is the figure to use notionally for the price of a game engine. Then there is software maintenance on that engine, usually about 33% of the cost of the engine, so add another \$100K per year. Let's bear in mind that the engine is good for about three years (until the next generation comes out), so in year four we see both the purchase of the next-generation engine and the software-maintenance fee for the old engine. And when we build on that licensed engine, we can't send the source code for our training simulation to anyone not licensed. So having chosen to license a commercial game engine to save time, we are stuck paying licensing forever.

The moral: if we are really to follow the path towards game-based simulation, DoD needs an open-source game engine yesterday. DoD also needs to consider open sourcing the painstakingly developed art within its games, so departments don't throw scarce resources at reinventing 3D soldiers, weapons, and training bases.

Development costs are the next line in the table. In the first year, we are building a lab (comprising computers and servers for the development team) and getting software tools installed. We are growing from zero staff towards, say, twenty-six. So in year one, we will spend about \$2M on the development team and setup. Year two has us spending \$2.5M for our team of twenty-six, plus management and admin costs. At twenty-four months, the game debuts on the Internet. In the case of *America's Army*, there were then four single-player levels and six multi-player levels (the complete release history through version 2.0.0a is

<u>Typical Costs</u>	<u>Year 1</u>	<u>Year 2</u>	<u>Year 3</u>	<u>Year 4</u>
Game Engine	\$300K	\$100K	\$100K	\$400K
Dev Costs	\$2.0M	\$2.5M	\$2.5M	\$2.5M
Operation Costs	\$1.5M	\$1.5M	\$1.5M	\$1.5M
Total	\$3.8M	\$4.1M	\$4.1M	\$4.4M

TABLE 1. Typical entertainment game costs (loosely based on *America's Army* costs).

presented later in this chapter), additional online releases and again *more* as we start the second version something like \$1M to \$2M at the we will eke by with a spartan staff \$2.5M, and so on.

Operations costs begin near the game, a marketing firm to build promotion. If we are building a training publicity cost, but we cannot get as complex an agenda as *America's Army* terrain box) is on the order of \$1M terrain, HLA networking, and cost

### The Tough Issue Is Team Building

What is the biggest challenge in building a visual-simulation or virtual-reality becomes a whole new proposition

If we were building a visual-simulation twenty-six programmers—and if we were in college, we would consider our team was a well-engineered training simulator. Engineer art is not immersive. Not the developer's most dreaded work and it only cost \$50." The ignorant seems superior, the scoring system is better. These comments are industry

Team building for game development will have, say, four game programmers taught, who can do scripts but make level designers and artists. The focus is practically no interest. What is the recommendation of persons you hire? artists and designers lack degree bushes and come up empty. Humanities should not be expected to build effective

Getting your team to function as a producer (EP) or creative director is the father figure for the group (programmer, lead artist, and lead game). The EP's job is to make the art and tool suite and maintains an art this cross-cultural, interdisciplinary that a game pops out after twenty



presented later in this chapter). Year three, we are adding new content for additional online releases and again spend some \$2.5M. We ought to be spending *more* as we start the second version of the game. We ought to bubble up in cost by something like \$1M to \$2M at the start of the third year. For this chapter, however, we will eke by with a spartan staff and not show such a bubble. Year four is again \$2.5M, and so on.

Operations costs begin near the start of the project, as we fund servers to host the game, a marketing firm to build booths for E3, and travel costs associated with promotion. If we are building a training system, we don't really incur substantial publicity cost, but we cannot get around server costs. So building a game with as complex an agenda as *America's Army's* (say, infantry-based combat in a small-terrain box) is on the order of \$2M to \$3M per year. Add in bigger pieces of terrain, HLA networking, and costs go up.

### The Tough Issue Is Team Building and Maintenance

What is the biggest challenge in building games? If you're coming at this from the visual-simulation or virtual-reality world, it's team building—which suddenly becomes a whole new proposition.

If we were building a visual simulation in the mid-1990s, we might hire twenty-six programmers—and if one of those programmers had taken an art class in college, we would consider ourselves good to go. What we would end up with was a well-engineered training simulation with displays that sport “engineer art.” Engineer art is not immersive. Nor is it engaging. It inspires the outsider to utter the developer's most dreaded words: “my kid's video game looks better than that and it only cost \$50.” The ignorant public will also point out that Game X's AI seems superior, the scoring system is way more thought out, and the networking is better. These comments are industry standards—your mileage may vary.

Team building for game development is different. In a team of twenty-six, we will have, say, four game programmers (perhaps two with CS degrees and two self-taught, who can do scripts but maybe not C++). The remaining twenty-two will be level designers and artists. The formal education of the designers and artists is of practically no interest. What is important is their demo reel showing past work, whether in school, game companies, or on their own. Of highest importance is the recommendation of persons you already hired and trust. Because many first-rate artists and designers lack degrees, traditional hiring procedures beat the wrong bushes and come up empty. Human-resource departments and program managers should not be expected to build effective game teams; insiders build these teams.

Getting your team to function pipeline-fashion is the job of the executive producer (EP) or creative director. He may be thirty—maybe younger—but he is the father figure for the group. Under the executive producer are a lead programmer, lead artist, and lead designer (for the story and presentation of the game). The EP's job is to make sure his team masters the selected game engine and tool suite and maintains an efficient resource-management system, and that this cross-cultural, interdisciplinary group behaves well enough and long enough that a game pops out after twenty-four months of concerted effort. Whiners are

### Year 3 Year 4

00K \$400K

5M \$2.5M

5M \$1.5M

1M \$4.4M

(*America's Army* costs).



culled. In the game-development community, exactly how to create this teamwork is widely understood.

Back to our goal of building training systems with such a team: we begin to perceive an incipient cultural challenge; namely, we will have to ensure that the game people and training people get along. Put military officers in charge of the project, and we have an extra dimension of fun and understanding. One group shows up at 11am in t-shirts and flip-flops. The other group comes in at 6am in uniform—but leaves at 5pm, while the gamers toil till midnight. This makes for a prickly cultural interface and requires patience and understanding. You can help things along by supplying the right management and keeping the program manager away from the development team.

### **AMERICA'S ARMY DEVELOPMENT PIPELINE**

To suggest the development process, we sketch the production of *America's Army* (*AA*). We then cover *AA* as a case history of what can be done in a given time through that process. In the industry, a game like *AA* is called a first-person shooter (FPS). This genre assumes that the game is rendered in real-time and the point of view is that of the player looking through the eyes of his character. To develop an FPS, skilled individuals are needed in some key positions.

#### **Positions and Duties**

**Programmer:** Programmers are the technical glue that holds everything together. They maintain the game engine, merge code updates, add features and tools, ensure hardware compatibility, identify and fix bugs, and integrate all content into the one package that users install on their machines. They interact with all other team members to weave strands of content into a final product. Without programmers, creating a game would be impossible.

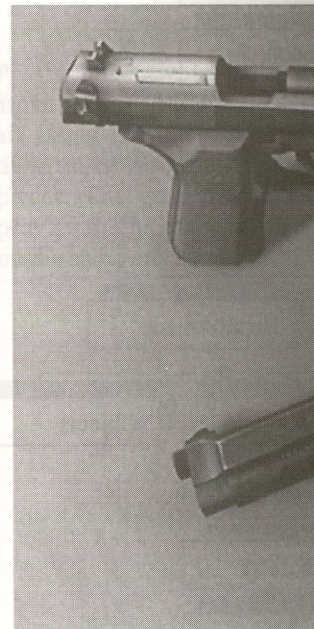
**Level Designer:** Level designers provide the biggest tangible piece of the game. Their job is to design and construct worlds in which the player can interact. They create terrain and buildings, place objects and sounds, add special effects, and, like stage managers, array each environment for its particular use. Level designers maintain frequent contact with everyone on the team.

**Artist:** Artists are responsible for the look and feel of the game. They create the surface, or "texture," of every wall, ceiling and floor, as well as flora, fauna, and faces. Artists typically develop the user interface and game icons and provide the artwork for special effects such as explosions, fire, water, smoke, muzzle flashes, lightning, etc. Generally speaking, if it can be seen in the game, an artist had something to do with it.

**3D Modeler:** While artists give life to the models, 3D modelers create the frameworks for them. They create the frameworks for everything from furniture to fire hydrants. They also create the game environments would be used to develop specialties. Two that p

**Character Modeling:** A character modeler creates the body proportion and structure of the characters. They have a good sense of bipedal locomotion and typically work with increased precision. Generally speaking, they will work throughout the course of development.

**Weapons Modeling:** The weapons modeler creates the animated weapons. Since weapons work in real-time (the weapons modeler works in real-time polygons) to ensure verisimilitude, they have little time for any other m



**PLATE 3.** Weapons, such as these



**3D Modeler:** While artists give you the skin, 3D modelers construct the bones. They create the frameworks for the artifacts that populate the game environment, from furniture to fire hydrants, phone poles to forearms. Without 3D modelers, game environments would be nothing but static, empty shells. Some 3D modelers develop specialties. Two that pertain to *America's Army* are as follows:

**Character Modeling:** A character modeler must have a highly developed sense of body proportion and structure to create realistic figures. They must also have a good sense of bipedal locomotion for realistic animation. Character modelers typically work with increased polygons, which adds extra complexity to their craft. Generally speaking, they will work primarily on characterization tasks throughout the course of development.

**Weapons Modeling:** The weapons modeler determines how each weapon will be animated. Since weapons will typically be the largest element on a user's screen, the weapons modeler works in minute detail (for which he has a large budget of polygons) to ensure verisimilitude. He will typically work on weaponry alone, with little time for any other modeling work.



PLATE 3. Weapons, such as these M-9 pistols, receive fine detail



**Sound Engineer:** The sound engineer creates, mixes, and imports into the game engine all the sounds the player hears. From bullets to footsteps to crickets, the sound engineer provides the hundreds (perhaps thousands) of effects that make an environment sound alive.

**Project Leaders:** Games can be highly complex, and the FPS is one of the more difficult genres to work in. Every good development team includes a number of project leaders, including a producer/director, lead designer, lead artist, and lead programmer. Depending on the size of team and the complexity of project, a small support staff will also be necessary.

**Core Game Components**

The following diagrams illustrate the core components of a typical FPS game. Figure 1 depicts a hierarchy of these components, while subsequent diagrams break down positions and interdependencies. Note that there is a deeper interdependency that cannot readily be depicted.

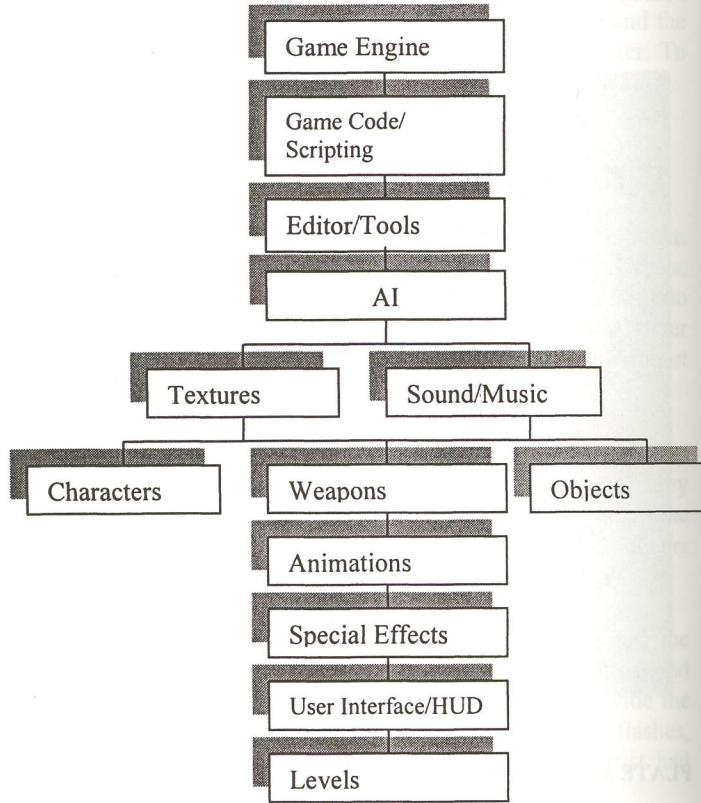


FIGURE 1. General Hierarchy of Core Components

At the foundation of every game is the game engine. The game will depend on this low-level department to ensure the game engine is important that this complex and properly. If the game engine fails, the

FIGURE 2. Game Engine

Programmers write game code and atmosphere and identity. Written on all assets into a coherent interactive department work together in a give-pieces. The game code and script provide the functionality that distinguishes on the same engine.

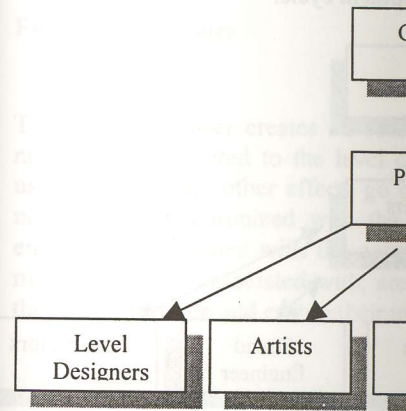


FIGURE 3. Game Code/Scripting



At the foundation of every game is the game engine (Figure 2). Every element of the game will depend on this low-level piece, and it is the task of the programming department to ensure the game engine can support the final product. It is extremely important that this complex and crucial element be maintained and organized properly. If the game engine fails, the project fails with it.

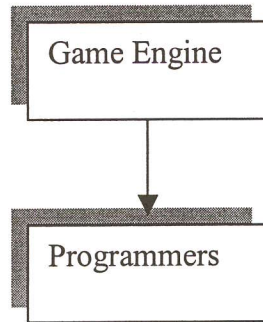


FIGURE 2. Game Engine

Programmers write game code and scripts to produce the game's peculiar atmosphere and identity. Written on top of the game engine, this code incorporates all assets into a coherent interactive experience. Programming and every other department work together in a give-and-take manner to successfully integrate the pieces. The game code and scripting set the scope of the overall design and provide the functionality that distinguishes your game from all other games based on the same engine.

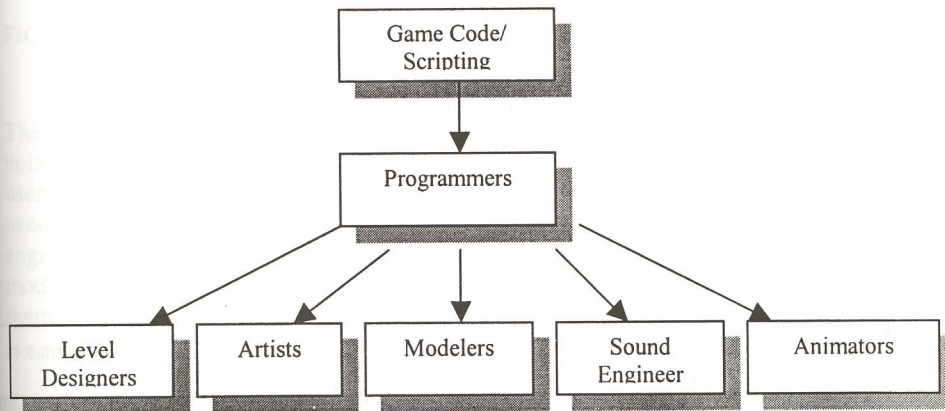


FIGURE 3. Game Code/Scripting



To facilitate the use of game code, the programming department provides the team with a game editor and tools for importing assets. Although these tools can be time consuming to create and maintain, ultimately they save countless man-hours and prevent bottlenecks by providing an assembly line for developing and integrating content. As the game evolves, so must the tools that support the team.

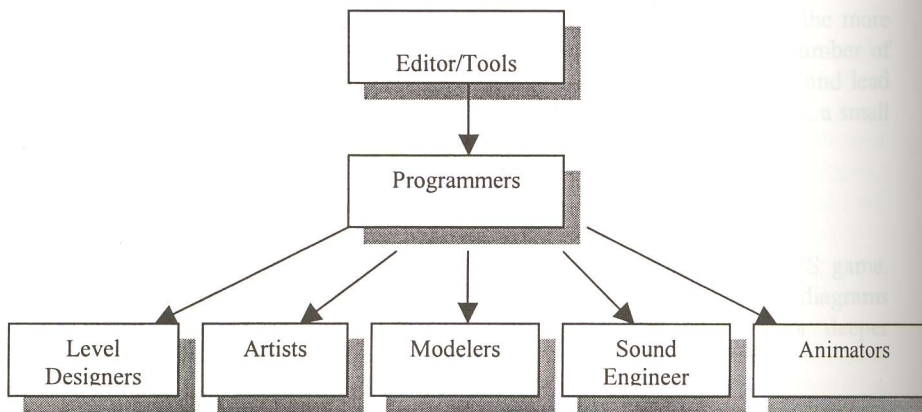


FIGURE 4. Editors/Tools

Artificial Intelligence (AI) is similar to game code, but more specialized and complex. To create AI, programmers work directly with each department of the team. For example, computer-controlled characters need an environment to run around in, so the programmers work with level design to ensure their proper setup. The art and modeling teams provide character models to attach the AI to, and the animator and sound engineer breathe life into these characters through movement and sound. Only when all these elements come together is AI fully functional in the game. It is typically a long process and requires one or more dedicated programmers through the course of the development cycle.

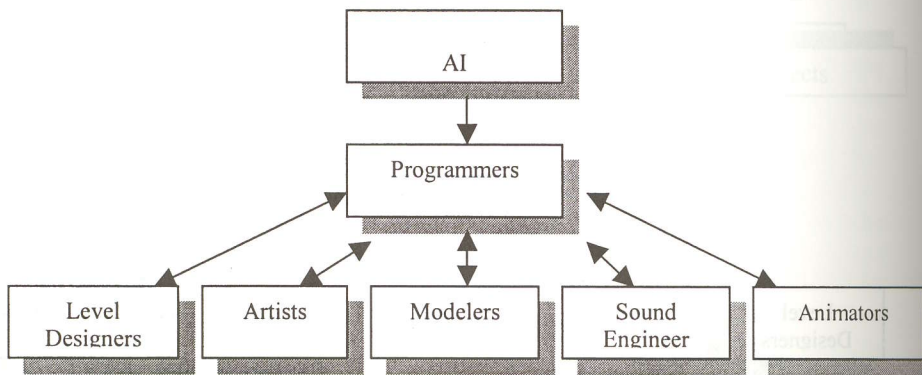


FIGURE 5. Artificial Intelligence

The artists provide texture maps for the walls, ceilings, and floors on all game objects (such as interface screens, and for all painted so that they wrap around environments require that a new paradigm. Texture maps are used for them, the characters, weapon

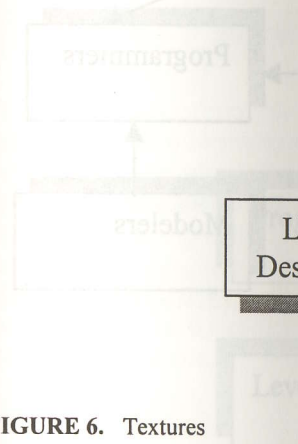


FIGURE 6. Textures

The sound engineer creates sound effects and noises are distributed to the user interface and other effects. The sound engineer need to be synchronized with the models they are associated with. The sound engineer collaborates with the modelers to ensure they are associated with them into the code and ensure



The artists provide texture maps to the level-design team, who then place them on the walls, ceilings, and floors of their game environments. Texture maps are used on all game objects (such as furniture, characters, weapons, etc.), in the user-interface screens, and for all in-game icons. For 3D objects, texture maps must be painted so that they wrap precisely around the model in a custom fit, while environments require that textures be painted according to a mathematical paradigm. Texture maps are essentially the visual matter of the game; without them, the characters, weapons and environments they cover would be invisible.

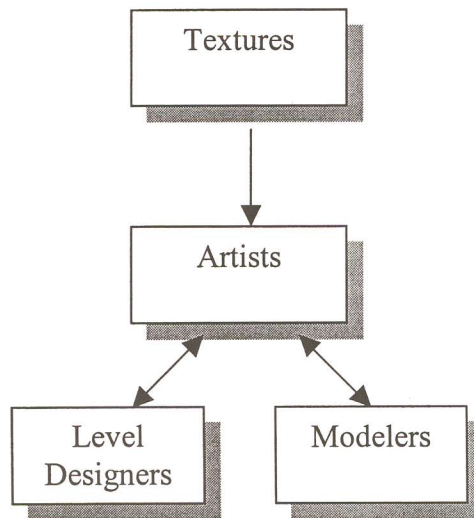


FIGURE 6. Textures

The sound engineer creates all sound and music files in the game. Background noises are distributed to the level designers for implementation. Sounds for the user interface and other effects go directly to the programming staff. Those that need to be synchronized with the movement of weapons and characters, the engineer collaborates with the animation team. These elements, along with the models they are associated with, are then given to the programmers, who import them into the code and ensure their unified functioning.



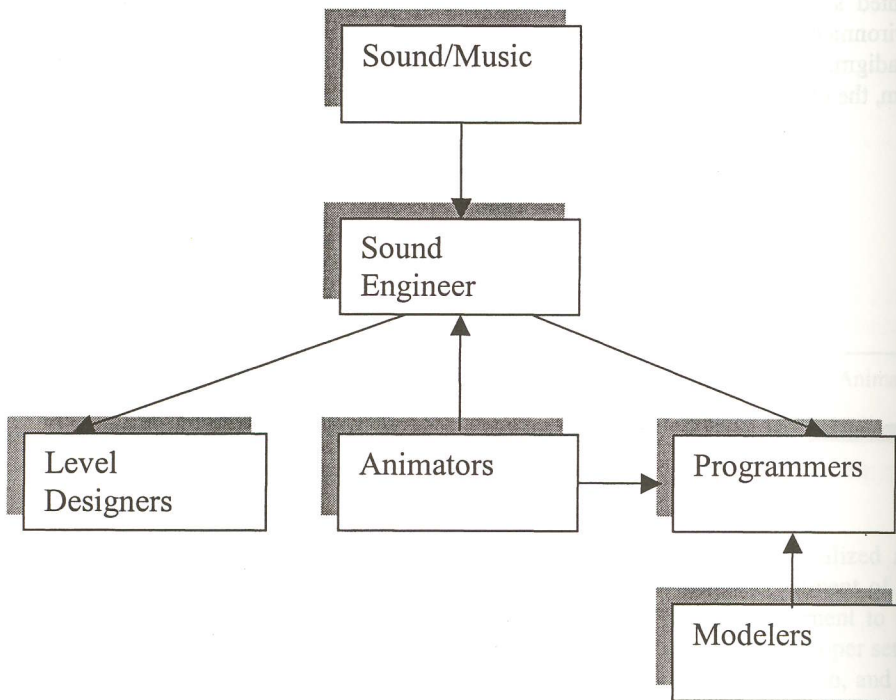


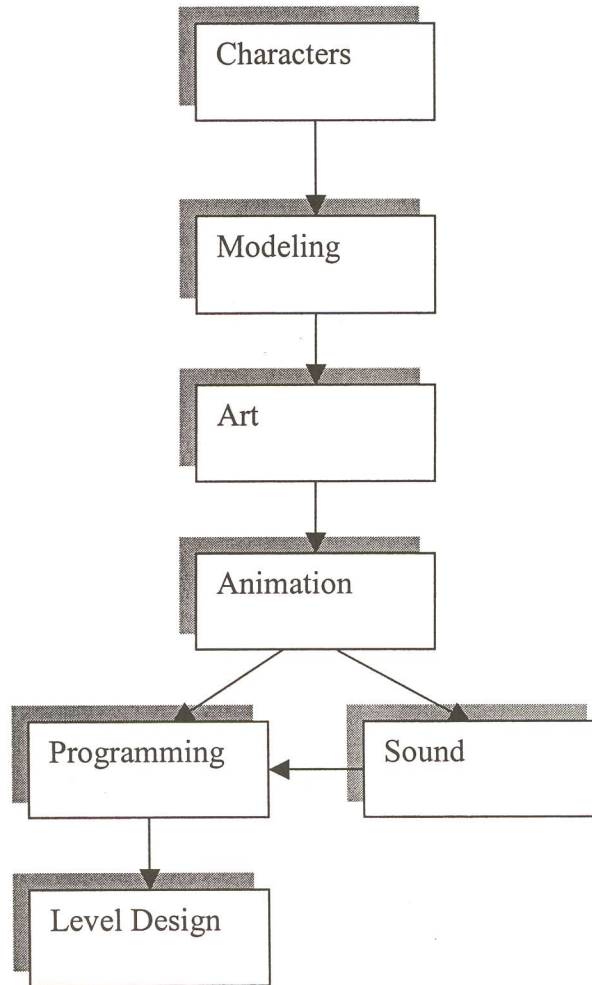
FIGURE 7. Sound/Music

Character models are created by a specialist 3D modeler then texture-painted by an artist (or the modeler, if he has the skill). When finished, the painted character is passed to an animator. Motion-capture data is applied to the object and hand tweaked. The completed object and animation data are sent to the programming team, who integrate it with the game code and attach any available AI functionality. The sound engineer then creates and synchronizes sounds for the character for addition by the programming team. Finally, the level-design team places the functional character into the game environments.

FIGURE 8. Characters

A specialized 3D modeler created, he or an artist paints and sets up the model, animates its incorporation into the game environment and the artists create elements and write game code. Level designers add the finished wea





**FIGURE 8.** Characters

A specialized 3D modeler creates weapons models. Once the model has been crafted, he or an artist paints a texture for it. It is then handed to an animator, who sets up the model, animates it, and sends the resulting data to programming, for incorporation into the game code. The sound engineer provides sounds for the weapon and the artists create special effects. Programmers then integrate these elements and write game code that defines the weapon's functionality. The level designers add the finished weapon to the game's environments.



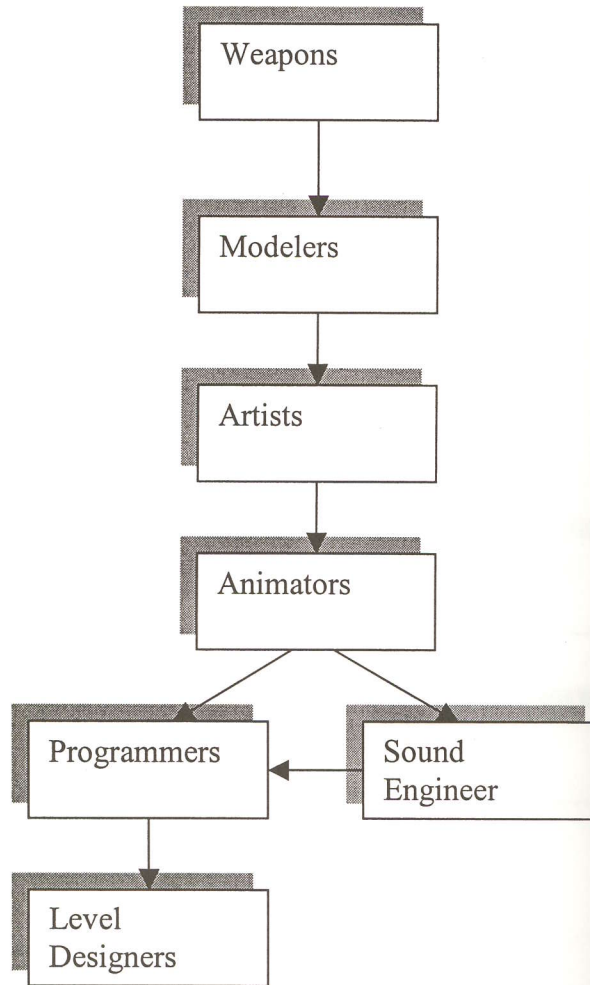


FIGURE 9. Weapons

3D Modelers create the world objects that are placed in the game environments: for example, light fixtures, vehicles, trees, grass, bushes, fences, and rocks. Once a world object has been created, an artist paints its texture map. The finished object is imported into the game code and placed in the game environments by level designers.

...the project...  
...the project...  
...the project...

FIGURE 10. World Objects

The animator determines the environment for the game. If AI is to be implemented, what animations are necessary. During a motion-capture session, in which these services must be contracted, the animator applies motion-capture to the modeler's objects, applies motion-capture to the modeler's objects, distributes the assets to programmers, and supplies audio to the programmer to the environments.

...the project...  
...the project...  
...the project...

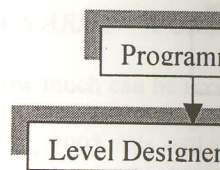


FIGURE 11. Animations



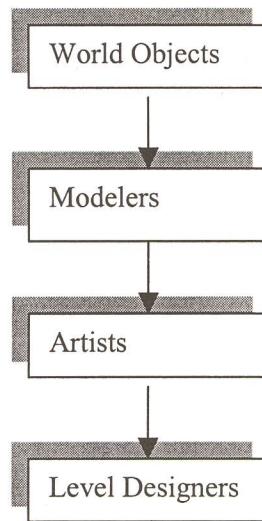


FIGURE 10. World Objects

The animator determines the entire range of motion for all moving elements in the game. If AI is to be implemented, character behaviors are examined to determine what animations are necessary. Once this has been decided, the animator directs a motion-capture session, in which an actor performs specified movements (usually these services must be contracted out, at high cost). The animator takes the modeler's objects, applies motion-capture information (adjusting as needed), and distributes the assets to programming and sound engineering. The sound engineer supplies audio to the programmers and level design adds the finished components to the environments.

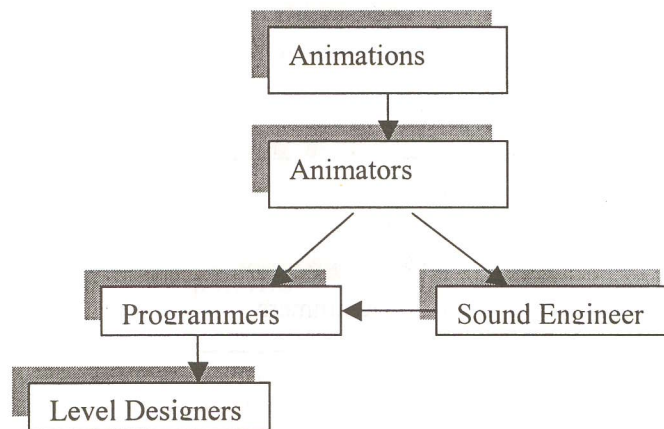


FIGURE 11. Animations

Special effects are an often-overlooked element that can be applied to virtually every aspect of the game, adding polish and interest. Clouds that pan across the sky, muzzle flashes, tracer fire, and water dripping from a leaky pipe are just a few of the effects that can make a game environment feel alive. These effects are usually created by the art team, who relay them either to level design for integration into the environments or to programming for placement directly into code. Typically, special effects are added towards the end of the project, when all other assets have been completed.

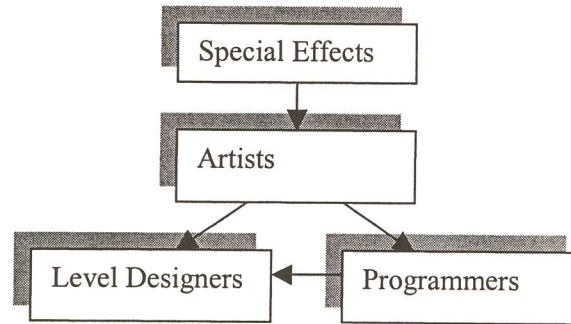


FIGURE 12. Special Effects

For the user to understand and play the game, a user interface and icons must be designed and implemented. These assets are typically created by the art department, who distribute them to the programming team for writing into code. Because the interface has to be updated as new features appear, it is important that it be robust and dynamic enough to grow as the game evolves.

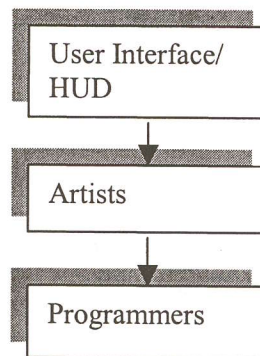


FIGURE 13. User Interface/Heads-Up Display

Level designers create the g...  
is served, it is here that all...  
final presentation. With th...  
team to ensure that each co...  
department is the hub for a...  
hub for all content. If the le...  
suffer.

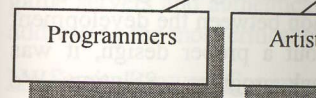


FIGURE 14. Game Environm...

## SUMMARY

There are many pieces to a g...  
battle, because it leads to a...  
design and project leaders...  
Scheduling – deciding which...  
the project meets its goals...  
above, there are many interd...  
and many risks: if one eleme...  
But with planning, good sta...  
overcome these risks and pro...

## AMERICA'S ARMY – A CA...

To show how much can be a...  
describes the *America's Army*...  
December 21, 2003. We wil...  
concerns associated with it...  
*AA's* development reveals is...



Level designers create the game environment. Like the dish on which a fine meal is served, it is here that all the components come together and the user enjoys the final presentation. With this in mind, the level designers work closely with the team to ensure that each component works as planned. Just as the programming department is the hub for all technical elements of the game, level design is the hub for all content. If the level-design team misses the target, the entire game will suffer.

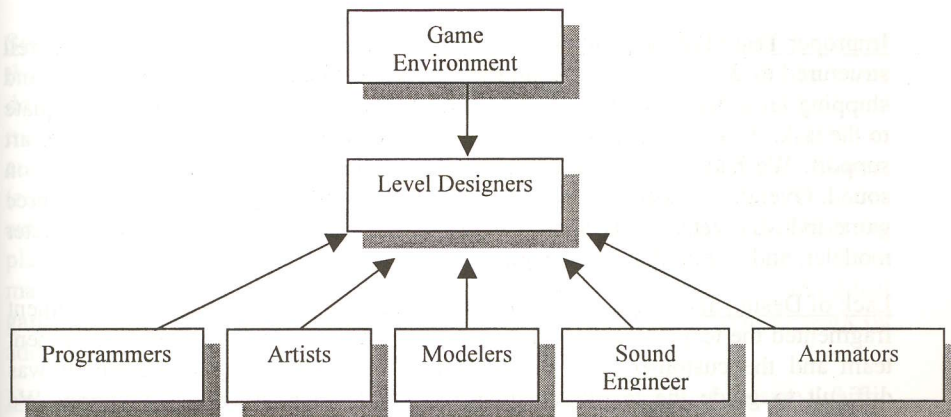


FIGURE 14. Game Environment

### SUMMARY

There are many pieces to a game like *America's Army*. Identifying them is half the battle, because it leads to a solid plan of action, which starts with good game design and project leaders who can communicate the design to the team. Scheduling – deciding which pieces are constructed when and by whom – helps the project meet its goals (yes, we're talking Microsoft Project). As illustrated above, there are many interdependencies among the components of an FPS game, and many risks: if one element fails, the ripples are felt throughout the enterprise. But with planning, good staffing and coordination, the development team can overcome these risks and produce a well-constructed, high-quality game.

### AMERICA'S ARMY – A CASE HISTORY

To show how much can be accomplished within three years, the following section describes the *America's Army* development from inception to the 2.0.0a release of December 21, 2003. We will enumerate the contents of each release discuss the concerns associated with it, and approximate the time spent. This close look at *AA's* development reveals issues that come up in developing large-scale games.



Some of the difficulties we encountered will doubtless crop up in your project. We hope our experience will prove useful.

### *America's Army Pre-Release – August 2001*

In August 2001, the *AA* project was seriously understaffed and unable to prosecute proper development. Major obstacles to success included the following issues:

**Improper Team Balance and Organization.** At this juncture, the team was not well structured to develop a first-person shooter. No one had experience in creating and shipping an actual product, and the team structure was inefficient and inadequate to the task. An overabundance of designers was coupled with a severe dearth of art support. We had a character animator, but no character modeler, and no one on sound. Overall, the team lacked cohesion and leadership. **Solution:** We hired three game-industry veterans as team leaders to rebalance the team, hired a character modeler, and acquired a sound engineer.

**Lack of Design and Common Vision.** The absence of a thorough design document fragmented the team's vision and precipitated confusion between the development team and the customer (i.e., the US Army). Without a proper design, it was difficult to guide the team, schedule tasks, and track progress. **Solution:** We focused on the overall mission statement, which was to develop a game with appeal similar to that of *CounterStrike*. To this successful model, we added heavy emphasis on realism and Army values and training.

**Technical Issues.** The game engine licensed for *America's Army* was still in development; in fact, during the entire course of development, the technology was constantly in flux. Many systems were not in place or inadequate for the game's needs, and completion of the engine was not anticipated until after the scheduled release of *AA*. Due to the development team's inexperience, the game's database structure was vastly inefficient and lacked consideration for distribution. Many of the game's assets were un-optimized or beyond the technical specifications of the game engine. Many of the steps and tasks necessary for success went unaddressed.

**Solution:** The engineering team wrote a number of new systems from scratch (approx 150,000 lines of code for the initial release of the game). We reorganized the art database and created a standard structure for all file formats and a team-wide methodology for database organization. Game assets were optimized to run well under the game engine. We cut a number of elements that were outside the engine's specifications. Task-management software was implemented to organize and track progress.

### *Version 1.0 Release – July 4, 2002*

The first version of *America's Army* was released on July 4, 2002. With the game a runaway success, the Army and development team were unprepared for the sheer

volume of players that flooded the game. The need for more servers was apparent as the public discussion forums and even prevented some servers from being delayed at launch so that the Army labeled the initial release as a "beta." It was understood it was really a "beta" with during this phase are a

**Server Overload.** Initially, the game. The average server downloaded over 500,000 players had to wait for authentication server that (required for multi-playing authentication server, too, for players to enter. Because so many dormant errors emerged from game servers and authentication servers, addressing the most critical of

**No Server-Browser/Community.** The in-game server browser was included with the download but there was no mechanism by which server-browser software to fix the problem of server overload and them. **Solution:** The development team well as packages for setting up

**Game-Play Bottleneck.** The complete the single-player (familiarization, and tactical). online and participate in a mission could be played. Additionally, winning team in the MOUT missions. While this seemed to be bottlenecks and yet another bottleneck not adequately describe the mission people were confused about the away with the online-training completing the single-player level

**Training-Level Bugs.** Both bugs. In the case of the rifle range to cheat and bypass qualification script prevented many players. **Solution:** The development team



volume of players that flocked to the game. Game servers were massively overloaded, and the need for a professional quality-assurance team became apparent as the public discovered critical bugs that detracted from the experience and even prevented some players from running the game. Several features had been delayed at launch so that the July 4<sup>th</sup> deadline could be met. Because of this, the Army labeled the initial launch of the game the "recon" version. Most players understood it was really a beta version. Issues that the development team dealt with during this phase are as follows:

Server Overload. Initially, the Army stood up only 140 servers for the launch of the game. The average server could accommodate 24 players. With the game downloaded over 500,000 times that weekend, the servers were swamped and many players had to wait days to play. Additionally, the game used an authentication server that validated players' having completed basic training (required for multi-playing) before allowing them onto a game server: this authentication server, too, was overwhelmed, making it even more difficult for players to enter. Because so many players had never played the game at once, many dormant errors emerged. Solution: The Army quickly stood up additional game servers and authentication servers. The development team went to work on addressing the most critical errors and applying server-side fixes.

No Server-Browser/Community-Server Support. At the release of version 1.0, the in-game server browser was not finished. As a stopgap, *Gamespy Arcade* was included with the download and was required to find and join game servers. There was no mechanism by which users could set up their own servers or use other server-browser software to find game servers. This shortcoming exacerbated the problem of server overload and irritated players by forcing unwanted software on them. Solution: The development team completed the in-game server browser, as well as packages for setting up user servers and user-created browser software.

Game-Play Bottleneck. The initial release of the game required that all players complete the single-player training courses (rifle-range, obstacle, weapons-familiarization, and tactical). Once these courses were finished, players had to go online and participate in a multiplayer training exercise before additional scenarios could be played. Additionally, until a user had played online and was part of a winning team in the MOUT McKenna training level, he could not proceed to other missions. While this seemed a good idea, in practice it created additional bottlenecks and yet another barrier to entry. To make matters worse, the game did not adequately describe the requirements for participation in further missions, so people were confused about what they were supposed to do. Solution: We did away with the online-training requirement and changed the game so that only completing the single-player levels was necessary.

Training-Level Bugs. Both the rifle range and obstacle course suffered critical bugs. In the case of the rifle range, players discovered an exploit that allowed them to cheat and bypass qualification. In the obstacle course, a logical error in the script prevented many players from finishing and proceeding with the game. Solution: The development team immediately fixed these problems.



**Multi-Player Bugs.** A number of critical bugs in the multiplayer portion of the game were discovered after initial release, ranging from graphical glitches to serious flaws in game play that marred the overall experience. In the case of the collapsed-tunnel mission, a logical flaw in the objective system miscounted victories and losses. In many cases, a victorious team was credited as having lost. This frustration led most users to avoid the mission. **Solution:** Identified the most severe problems and began working on fixes.

#### **Version 1.0.1 Release –July 12, 2002**

Version 1.0.1 of *America's Army* was released on July 12, 2002. As implied by the version number, this was a minor release, consisting primarily of a patch for the worst problems of version 1.0. The main issues addressed were as follows:

- Corrected client and server-flooding issues. This fix stabilized servers that were overloaded by network traffic.
- Fixed training level bugs. These fixes addressed the most critical issues involving the rifle range and obstacle course.
- Added community game-server support. This allowed the use of alternative server browsers for finding game servers.
- Added a dedicated server executable, allowing players to stand up their own game servers.

#### **Version 1.1.1 Release –August 1, 2002**

On August 1, 2002 the development team released version 1.1.1, the "marksmanship pack." This release added the Army's sniper schools and the M24 and M82 rifle positions to the game, features originally scheduled for the initial July 4<sup>th</sup> release, but fallen behind schedule. Eligibility to play the marksmanship levels was based on scores from the original rifle-range training level. A player who shot 36 out of 40 targets in the final test could try to qualify as a sniper. Only those players who passed the marksmanship training levels could take a sniper position in online play.

While finalizing this version, an unfortunate database error was discovered: the authentication server was logging only pass/fail results for the rifle range. Once a player was determined to have passed the course (with a score of 23 or above), the authentication server did not bother to record subsequent attempts, so that players who had met the basic qualifications could not return to the rifle range and try for better scores so they could move on to sniper school. In the end, we reset the rifle-range scores for all players to force the necessary changes to the authentication system. Many players who had already qualified for sniper (an extremely difficult feat) found they were obliged to qualify again. This naturally had a very negative impact on the player community.

To make matters worse, positions were filled, meaning any time. With the release of sniper. Virtual fratricide between sniper rifles. Needless to say, brainstorm ways to curb it. During this release, training requirement. Iron through the painful launch badge of honor. Many felt should too. Regardless, server bandwidth.

Other changes in this

- Added idle-player kick, limited server space, playing (to preserve the annoyed those in the game was added to time i Occasionally players additional fixing in su
- An in-game server browser rudimentary functionality necessity of using *Game*
- Added MILES grenade Army uses for training based on MILES scenario on the development team add a MILES-equipped addition by the commu
- The development team amber to green.

#### **Version 1.2.0 Release –August 22, 2002**

Released on August 22, 2002, airborne and ranger schools training levels that depicted ranger school offered no training to take place online McKenna training scenario requirements was determined were converted to standard nothing players had to do to



To make matters worse, *AA* opened the sniper role only after other team positions were filled, meaning there were only a few sniper positions available at any time. With the release of the marksmanship pack, *everyone* wanted to be a sniper. Virtual fratricide broke out as people killed team members just to steal their sniper rifles. Needless to say, we did not anticipate this abuse and had to brainstorm ways to curb it.

During this release we also did away with the MOUT McKenna online-training requirement. Ironically, this caused an outcry from those who had gone through the painful launch experience and saw completion of the training as a badge of honor. Many felt that since they went through MOUT McKenna, others should too. Regardless, it was necessary to remove the requirement to free up server bandwidth.

Other changes in this release included:

- Added idle-player kick. In the initial release, it was discovered that with the limited server space, many players neglected to log off when they weren't playing (to preserve their slot). This infuriated players who couldn't get in and annoyed those in the game who saw a team member just standing there. A fix was added to time idle players and kick them off after a certain period. Occasionally players who were not idle would get the boot, requiring additional fixing in subsequent releases.
- An in-game server browser was finally added. While offering only the most rudimentary functionality, it at least appeased players and removed the necessity of using *Gamespy Arcade*.
- Added MILES grenades to MILES missions. MILES is a laser-tag system the Army uses for training. At the Army's request, a number of *AA* missions were based on MILES scenarios (the irony of simulating a simulation was not lost on the development team). With release of version 1.1.1, the Army wanted to add a MILES-equipped grenade to these missions. Opinions concerning this addition by the community were mixed.
- The development team was asked to change the tracers of enemy fire from amber to green.

#### ***Version 1.2.0 Release –August 22, 2002***

Released on August 22, 2002, version 1.2.0, the airborne/ranger pack, introduced airborne and ranger schools to the game. While the airborne school came with two training levels that depicted an abridged version of the Army's actual training, the ranger school offered no training levels at all. The original design called for ranger training to take place online with other players, but after the debacle of the MOUT McKenna training scenario, adding another round of multiplayer training requirements was determined not worth the risk. Instead, the ranger-training levels were converted to standard online scenarios. The disadvantage was that there was nothing players had to do to qualify for these maps. In the end, we required that all



other training be completed before ranger maps could be played. While this was a workable compromise, it clashed with existing paradigms in the game.

Other problems encountered with this release revolved around the airborne portion of the game. The technology used for *AA* was not ideal for simulating flight, and the artists had to depend heavily on tricks to create the illusion of parachuting. While this worked well in the single-player training missions, where the experience could easily be constrained, multiplayer missions posed hurdles and challenges that were never fully resolved. Parachuting introduced a host of bugs, not to mention heavy demands on the processor. While ultimately the team this feature adequately, associated problems plagued the entire production cycle, including parachutes not opening (and players falling to death), parachutes deploying inside planes, parachutes stuck on the body after landing, players stuck together or stuck on other objects, players unable to move after landing, and a host of related technical issues. Although this was only a small feature in the game, it consumed a great many man-hours.

Additional highlights for version 1.2.0 included to following.

New Voice-Overs for Radio Commands, Shouts and Whispers: During development, team members and Naval Postgraduate School students were often used as voice actors for the game. While this saved the cost of hiring professionals, it meant that creating good voice-overs (VO) was a struggle. A particularly good reader might be a military officer, stationed at NPS for only a short time, or an original reader might no longer care to participate. When this happened, a new VO candidate had to be located and the entire voice-over sequence recreated. Moreover, voice files tend to be quite large, and the continual changes frequently increased the download size of subsequent releases. This aspect of development proved frustrating, an ever-changing facet of the game.

Adjusted Team-Balance System: In multiplayer games, it is customary to include team balancing. If one team heavily outnumbers the other, the system will shuffle players to achieve equity. Also, if one team consistently beats another by large margins, the system will exchange players to make the teams equitably matched. While this sounds good in theory, it can create problems. Players may not understand the computer's arbitrarily changing the conditions of the game, and the system itself may respond to very specific contexts only. Without a professional QA department, many of the flaws in the auto-balancing system aren't discovered until after a new version of the game is released and feedback is received from irritated players. In the case of *America's Army*, this feature was adjusted several times before it was deemed acceptable. In all likelihood, it was never truly perfected and there are still players who are not satisfied with it.

Adjusted Vote-Kick Feature: The vote-kick system was created so that players themselves could enforce the rules of the server. If an unruly player were causing havoc, a player could call for a vote to kick him off the server. While this is a common tool in multiplayer games, we didn't foresee the ways in which it might be abused. We found that many players were being tossed for reasons outside the

scope of the system. Like vote-kick numerous times. human behavior, so a perfect

Adjusted Weapon Distribution: to select their weapons, but the accompanying weapons. The weapon-distribution dispersed among players. personal-weapon preference favored weapon; but the system provide consistent results. The constant modification by the

Three New Multi-Player Missions to the game: the F Because we had few testers allow us to test maps with appeared with these levels. level. It was found that if so mission began, the round in servers frequently, this level removed from server rotation

#### **Version 1.2.1 Release –August**

On August 24, 2002, only two created to deal with the Specifically, several fixes were system to make the mountain

#### **Map Pack Release –October**

On October 3, 2002 the development missions for the game: JRTC finished for some time, but would be used for strategic-marketing through standard distribution Army recruiters only. After the next release. Although this patch development team several days new release so that players had full version again. With the release for two different versions of and apply the patch according



played. While this was a  
in the game.

ved around the airborne  
not ideal for simulating  
to create the illusion of  
training missions, where  
missions posed hurdles and  
introduced a host of bugs,  
ultimately the team this  
entire production cycle,  
g to death), parachutes  
ter landing, players stuck  
e after landing, and a host  
all feature in the game, it

ng.

and Whispers: During  
chool students were often  
st of hiring professionals,  
ggle. A particularly good  
only a short time, or an  
this happened, a new VO  
ver sequence recreated.  
tinal changes frequently  
is aspect of development

is customary to include  
er, the system will shuffle  
ly beats another by large  
teams equitably matched.  
blems. Players may not  
tions of the game, and the  
y. Without a professional  
g system aren't discovered  
feedback is received from  
ature was adjusted several  
hood, it was never truly  
with it.

as created so that players  
nrulely player were causing  
he server. While this is a  
he ways in which it might  
ed for reasons outside the

scope of the system. Like the team balancing system, it was necessary to adjust  
vote-kick numerous times. It's difficult for a computer to identify and regulate  
human behavior, so a perfect solution to game pests was never truly achieved.

Adjusted Weapon Distribution: In the original version, players were not allowed  
to select their weapons, but instead chose what role they wanted and were given  
the accompanying weapons, based on the actual structure of Army infantry units.  
The weapon-distribution system regulated how the various weapons were  
dispersed among players. The problem was that most players maintained a  
personal-weapon preference and wanted to find out what to do to obtain the  
favored weapon; but the system relied on mathematical voodoo that did not always  
provide consistent results. The result was great confusion among the players and  
constant modification by the development team.

Three New Multi-Player Maps: Version 1.2.0 added three new multiplayer  
missions to the game: the FLS assault, the swamp raid, and the mountain ambush.  
Because we had few testers at this point (as well as an internal network that did not  
allow us to test maps with a full contingent of players), a host of new problems  
appeared with these levels. The most dramatic involved the mountain-ambush  
level. It was found that if someone changed teams and then left the server after the  
mission began, the round immediately ended. With players entering and leaving  
servers frequently, this level was in effect unplayable and was temporarily  
removed from server rotation.

#### *Version 1.2.1 Release –August 24, 2002*

On August 24, 2002, only two days after the release of version 1.2.0, a patch was  
created to deal with the critical errors introduced in the previous release.  
Specifically, several fixes were made to new missions and to the team-structure  
system to make the mountain-ambush level playable.

#### *Map Pack Release –October 3, 2002*

On October 3, 2002 the development team released a map pack including two new  
missions for the game: JRTC Farm and Weapons Cache. These two maps had been  
finished for some time, but were delayed by request of the Army so that they could  
be used for strategic-marketing purposes: before releasing them to the public  
through standard distribution channels, these missions were first available through  
Army recruiters only. After a time of exclusivity, the missions were added to the  
next release. Although this practice seemed straightforward, it actually caused the  
development team several distribution problems. Patches were created with every  
new release so that players had only to download the new rather than retrieve the  
full version again. With the map pack however, our engineers now had to account  
for two different versions of the game (one with the new missions, one without)  
and apply the patch accordingly. Since this map pack fell outside the scope of the



team's normal distribution methodology, extra engineering was required to ensure that all players would be able to update the game seamlessly for the next release.

### *Version 1.3.0 Release –October 10, 2002*

Released on October 10, 2002, this version of *AA* added a host of new features, bug fixes, and adjustments. Since the game's initial release, the development team had been scrambling to finish uncompleted features. With version 1.3, they were finally able to consider the initial launch finished and begin focusing on new features and adjustments based on user feedback. While this release offered only one new multiplayer level (the mountain-pass arctic mission), great effort was put into improving the game overall. Some of the changes made in this release are as follows.

Combat Effectiveness Meter (CEM): Because *America's Army* attempted to portray a realistic combat system, there were a number of factors that could affect a player's accuracy and effectiveness while engaging the enemy, including posture (standing, crouching or kneeling), movement (e.g., running versus walking), use of weapons' iron sights, scopes, and bipod supports, and proximity to team leaders. While this allowed for a system more closely resembling the experience of real combat, the calculations were done behind the scenes, and players often were confused about the variance of weapon accuracy in the game. In version 1.3, a meter was added to the player's screen, resembling the equalizer bar on a stereo system: the higher the bar, the more effective the player in combat. As the player moved (for example, changed posture and speed), the bar rose or fell to reflect the effectiveness of the player's actions. This feature brought the inner workings of the combat system to the fore, allowing better understanding of how to be effective and what might cause poor performance.

Honor System: For some time, the Army had been looking for the development team to provide players with a comparative statistic showing accomplishment within the game. Version 1.3 answered this desire by adding an honor system. The honor system attached a persistent score (between 1 and 100) to every player. By tracking points scored against points lost, players could build their honor score and wear it as a badge for all to see.

Inevitably, however, many players wanted the score to reflect actual ability, rather than simple time invested in the game. Moreover, the honor system created a distinction between official and unofficial game servers, because only experience racked up on official servers was counted towards honor gain (to prevent exploitation of the system). This caused players to avoid unofficial servers and play on Army-sponsored servers only, hampering the growth of the game community. Over the course of the project, there were also several bugs and situations that could cause honor scores to be lost or reset, precipitating an outcry. While the development team made many alterations to the honor system, its full potential was never achieved.

Auto Weapon Low  
the player's weapon wo  
response, a system was  
his weapon automatica  
players found that their  
needed. These glitches

"Hit the Dirt" Feature:  
perform a combat dive v  
was well received, it wa  
stuck in level geometry  
the problem, many playe

Night Vision for Spectat  
out of the action and ma  
by viewing a particular t  
see the action in the low  
spectators and camera po

Adjusted Server Browse  
server browser so that pl  
participate on. More opt  
server browser.

Adjusted M249 Fire Mo  
players had learned to tap  
range weapon. This wa  
adjustments were made to

Adjusted Weapon Accur  
fired with increased realis

Adjusted Prone Moveme  
provide better performan  
certain actions.

Adjusted Footstep Volum  
were too hard to hear. The

Adjusted Sniper Rifle A  
accuracy system, so that s  
was targeted, but decreas  
through greater wavering i

Adjusted Grenades: It se  
adjusting and balancing  
discovered that a realistic  
leading to constant rebalan  
following changes were ma



ing was required to ensure  
sly for the next release.

and a host of new features,  
use, the development team  
with version 1.3, they were  
d begin focusing on new  
e this release offered only  
ision), great effort was put  
made in this release are as

*America's Army* attempted to  
of factors that could affect  
e enemy, including posture  
ing versus walking), use of  
proximity to team leaders.  
ing the experience of real  
s, and players often were  
e game. In version 1.3, a  
e equalizer bar on a stereo  
er in combat. As the player  
ar rose or fell to reflect the  
ht the inner workings of the  
ing of how to be effective

oking for the development  
e showing accomplishment  
dding an honor system. The  
nd 100) to every player. By  
build their honor score and

ore to reflect actual ability,  
er, the honor system created  
ers, because only experience  
s honor gain (to prevent  
avoid unofficial servers and  
the growth of the game  
were also several bugs and  
reset, precipitating an outcry.  
to the honor system, its full

**Auto Weapon Lowering:** In early releases, it was discovered that occasionally the player's weapon would penetrate level geometry and give away his position. In response, a system was modified so that when a player was too close to an object, his weapon automatically lowered to avoid it. While this solved one problem, players found that their weapons did not always return to proper position when needed. These glitches were addressed in subsequent releases of the game.

**"Hit the Dirt" Feature:** The new version of the game gave players the ability to perform a combat dive while running, quickly hitting the ground. While the feature was well received, it was eventually scaled back because players were sometimes stuck in level geometry after performing the maneuver. While scaling back solved the problem, many players were disappointed by the changes.

**Night Vision for Spectator Mode:** In *America's Army*, once a player is killed he is out of the action and may watch the game from a number of spectator cameras or by viewing a particular team member. In night missions, spectators often couldn't see the action in the low lighting. To compensate, night vision was provided to spectators and camera points.

**Adjusted Server Browser:** More detailed player and game info was added to the server browser so that players could better select the game servers they wanted to participate on. More options were also provided to sort the data received in the server browser.

**Adjusted M249 Fire Model:** In previous versions, it was discovered that many players had learned to tap the fire key of the M249 to turn it into a powerful, long-range weapon. This was at odds with the weapon's real performance, so adjustments were made to add variance to the burst-fire capabilities of the weapon.

**Adjusted Weapon Accuracy System:** We made adjustments so that all weapons fired with increased realism in shot patterns and bullet spread.

**Adjusted Prone Movements:** Movement in the prone position was adjusted to provide better performance over terrain and more flexibility when performing certain actions.

**Adjusted Footstep Volume:** It was discovered in previous versions that footsteps were too hard to hear. The volume was turned up to give better immersion.

**Adjusted Sniper Rifle Accuracy:** Adjustments were made to the sniper-rifle accuracy system, so that shots fired always hit the exact spot where the crosshair was targeted, but decreased combat effectiveness was translated to the player through greater wavering in the weapon's scope.

**Adjusted Grenades:** It seemed that the development team would forever be adjusting and balancing the way grenades were depicted in the game. We discovered that a realistic grenade does not necessarily equal a fun experience, leading to constant rebalancing and enhancing of the feature. In version 1.3, the following changes were made to the grenade system:



- Auto Grenade Notification: Many players were dying from grenades because they were unaware that they had been thrown. The development team added a feature whereby throwing a grenade triggered an audible warning to other players in the area. To reward stealth, the warning could be overridden if players moved slow in lobbing a grenade.
- Auto Weapon Switch Upon Grenade Throw: Many players were dying after throwing a grenade because they couldn't raise their weapons in time afterwards to defend themselves. We added automatic switching back to the primary weapon after a throw. Realizing that some players might dislike the feature, we included a menu option for disabling.
- Grenade Spin: In previous versions, grenades did not observe physics and traveled in a frozen position. For better realism, spin was applied.
- Dive on Grenades: The ability to dive on grenades was added, thus letting players save buddies from harm. Unfortunately, because of game perspective, it was difficult to judge exactly where to land. It turned out to be rare for anyone to exploit this ability; the feature was mostly ignored.
- Grenade Physics by Material Type: Changes were made so that grenades would react differently depending on the type of surface they encountered. Like the grenade spin, this increased apparent realism.
- Adjusted Variance of Fuse Length: Originally, all grenades possessed the same length fuse. We became aware that players had learned exactly how long they could hold a live grenade before throwing it, pulling off precision attacks that would not be possible in the real world. To compensate, the development team varied the fuse length, making accurate judgment impossible. From version 1.3 on, if players held on to live grenades, they risked blowing themselves up.
- Adjusted Auto Roll Distance: We adjusted how far grenades could be rolled.

#### *Version 1.4.0 Release –November 25, 2002*

Released on November 25, 2002, version 1.4 of *America's Army* was a minor release that offered one new mission (River Basin), and a handful of new features and bug fixes.

New Scoring System: A new scoring system de-emphasized killing the enemy and rewarded acting as a team and completing objectives. While hard-core gamers did not immediately embrace the system, many players found they were able to achieve higher scores without necessarily using violence. Ultimately, this created a more balanced experience while simultaneously improving the marketing message the Army sought to express.

"Report In" Feature: Based on player location and report their location was used by the development team to make sure they were in the game.

Binoculars: Team leaders were able to see and coordinate with team members.

Movement With Iron Sights: When using iron sights of a weapon, any movement was from a first person perspective. With version 1.4, players could look through the sights.

News for Login Screen: A new news screen for the Army could make general announcements.

Adjusted Automatic Weapon: When a player switched from standing to kneeling, the weapon would continue firing. This was a need to the attention of the development team.

Fixed Multiple Login Exploit: Fixed an exploit that allowed machines to login to different servers simultaneously. This was an unacceptable rate. To address this, we added authentication servers to check for duplicate logins on the server.

Another Grenade Adjustment: Added a feature that if the player pressed the fire button while a grenade was made available with the pin already lit.

#### *Version 1.5.0 Release –December 23, 2002*

On December 23, 2002, the development team released version 1.5.0 of *America's Army*. At this time, the game had come under more scrutiny for violence in video games. As the game became an irresistible target. The development team made modifications to counter the negative publicity. The elimination of the word "sniper" from the game (at several levels and weapon names) was one change. (Parental control marksmanhood schools). Parents were able to monitor language, weapon usage, and other content. These changes were designed to address parental concerns and letting parents control content.

In addition to parental controls,



“Report In” Feature: Based on user feedback, players’ ability to hit a single key and report their location was added. This well received feature required the development team to make substantial adjustments to, and testing of, every level in the game.

Binoculars: Team leaders were provided with binoculars to better scout positions and coordinate with team members.

Movement With Iron Sights: In previous versions, if the player was using the iron sights of a weapon, any movement would drop him to the normal weapon perspective. With version 1.4, players could move (albeit very slowly), while looking through the sights.

News for Login Screen: A news section was added to the login screen so that the Army could make general announcements about the game.

Adjusted Automatic Weapon Fire System: Adjustments were made so that if a player switched from standing to crouching while firing an automatic weapon, the weapon would continue firing during the posture change. Players had brought this need to the attention of the development team.

Fixed Multiple Login Exploit: It was discovered that players were using multiple machines to login to different game servers under the same account. By playing simultaneous games with one account, players were building their honor score at an unacceptable rate. To address the issue, the development team caused the authentication servers to check for multiple logins and kick offenders from the server.

Another Grenade Adjustment: To increase grenade realism, a change was made so that if the player pressed the fire button while selecting a grenade, the grenade was made available with the pin already pulled and ready to throw.

### *Version 1.5.0 Release –December 23, 2002*

On December 23, 2002, the development team released version 1.5. Around this time, the game had come under fire by a Miami attorney on a crusade against violence in video games. As the funders of AA, the U.S. government proved an irresistible target. The development team was required to make several modifications to counter the negative press generated by this man, including the elimination of the word “sniper” from the game (which involved major changes to several levels and weapon systems, as well as new voice-overs for the marksmanship schools). Parental controls were added so that parents could monitor language, weapon usage, and mission types, and limit displays of blood. These changes were designed to differentiate AA from most commercial games by letting parents control content.

In addition to parental controls, other changes in this release included:



Weapons-Cache Special-Edition Map: One of the most popular levels in the game was the weapons-cache mission. Many fans pointed out flaws in the map, as well as desired improvements. Based on this feedback, a new version of the mission was created, effectively doubling its scope. These changes were applauded, and the mission remains one of the most popular to date. By implementing improvements per popular demand, the team was able to foster goodwill and to assure the community of their voice in the game's evolution.

New Enemy Voices: With the help of the Defense Language Institute, the development team created a fictive enemy language, based on a combination of natural languages. Voice-overs of foreign students were recorded to create realistic shouts and enemy radio commands while ensuring that no speakers of an actual foreign language would be depicted as enemies of the United States. As a bonus, because the enemy language had its roots in reality, players found they could learn and understand the commands issued by opposing forces.

Optional Reason for Vote-Kick System: Previous versions revealed that the vote-kick system was inadequate because players were often in the dark as to why a player had called to ban another player. An optional reason was added so that when a player called a vote, the others could see why.

Army Star to Player Listing: The development team added to the scoreboard the ability to show whether a player was an active member of the US Army (subject to verification). When a verified soldier played in the game (and there were many of them), an Army star appeared next to his name. This allowed the community to know when they were interacting with actual soldiers and strengthened camaraderie between military and civilian players.

ROE Penalty Adjustments: Whenever a player injured or killed a team member or performed specific detrimental actions in the game, he suffered a penalty to his score for violating the "rules of engagement" or ROE. While this was an effective way to enforce Army values, the development team often found it necessary to tweak the system to ensure proper play balance.

Server-Browser Adjustments: Adjustments were made so that the server browser distinguished between leased servers and official servers. How many LAN servers could be displayed at a time was also increased.

### **Version 1.6.0 Release –March 16, 2003**

Version 1.6 of *America's Army* was released on March 16, 2003. This release took considerably longer to complete than previous versions due to an update of the game's core technology: Epic Games, who created the software *AA* was built on, had released a major update to the game engine. The development team had to merge the updated technology with the game's current code base. After the months of work had been put into the game, there were vast differences between the code base and that update. The merger took about six weeks of programming, as well as

a number of weeks to ad  
was a requirement if *Ame*  
Although only one r  
level was the largest map  
technological limits, and  
playing experience. Fla  
discovered, and it was fo  
load the level that the au  
from the server. A numbe  
released.

Other changes in version 1

Projectile Penetration: Pr  
was blocked and consider  
bullets passed through per  
and force (depending on th  
This yielded a dynamic ch  
served as cover could no lo

Projectile Ricochets: The  
angles was introduced for r

Bullet Decals on Static ar  
bullets to leave marks on s  
it also increased processor  
settings were added to cont

New Sound Effects: New s  
on concrete and carpet.

New Texture-Detail Optio  
enabled players to adjust tex

Password-Entry Window t  
servers and control access t  
browser for passwords.

Spam Control for Messagin  
flooding the in-game messa  
play. To compensate, the er  
be sent by a player in a give

New Desert Camouflage:  
Army had changed its dese  
were changed accordingly.

New Loading Screen: A ne  
engine was tied up with load



a number of weeks to adjust content to work with new features. While painful, it was a requirement if *America's Army* were to keep its cutting edge.

Although only one new mission accompanied this release, the radio-tower level was the largest map the development team had created. This mission pushed technological limits, and frequent adjustments were made to reach a smooth playing experience. Flaws in the authentication and loading systems were discovered, and it was found that low-end machines were taking so much time to load the level that the authentication server would time them out and drop them from the server. A number of band-aids were applied before this version could be released.

Other changes in version 1.6 included:

Projectile Penetration: Previously, any time a bullet struck an object, the bullet was blocked and considered spent. Version 1.6 introduced penetration, by which bullets passed through penetrable objects and continued with diminished velocity and force (depending on the material hit) as well as condign entry and exit effects. This yielded a dynamic change in game play, because objects that had previously served as cover could no longer be depended on.

Projectile Ricochets: The tendency for bullets to ricochet when fired from certain angles was introduced for more realistic ballistics and added tension.

Bullet Decals on Static and Dynamic Objects: The technology update allowed bullets to leave marks on static and moving objects. While this increased realism, it also increased processor overhead. To avoid sluggishness in low-power machines, settings were added to control how many bullet marks could be displayed at once.

New Sound Effects: New sounds were added for ricochets, as well as for footsteps on concrete and carpet.

New Texture-Detail Options: An array of new settings in the menu system enabled players to adjust texture detail to suit the power of their machines.

Password-Entry Window to Server Browser: To allow users to set up private servers and control access to them easily, a new window was added to the server browser for passwords.

Spam Control for Messaging System: It had been discovered that players were flooding the in-game messaging system, effectively ruining communication during play. To compensate, the engineering team controlled how many messages could be sent by a player in a given time.

New Desert Camouflage: During this development period, we learned that the Army had changed its desert-uniform camouflage. Desert uniforms in the game were changed accordingly.

New Loading Screen: A new loading screen was added to indicate when the game engine was tied up with loading new content into memory.



Fatigue Element to Jumping Abilities: Many players were demonstrating a tendency to jump up and down in the game, a term known to gamers as “bunny hopping.” Since soldiers are typically weighted down with equipment, such action was not in keeping with the degree of realism we were attempting to portray. Fatigue was therefore added so that repeated jumping caused the player’s character to tire and be unable to continue.

Grenade Aiming: Players found that, because of the perspective in the game, aiming a grenade accurately was extremely difficult, requiring a great deal of guesswork. To make the system more intuitive, the player’s onscreen hands were changed so that the gap between the forefinger and thumb of the lead hand was positioned over the center of the screen, enabling the player to use it as a guide.

Improved Weapon-Jam System: The algorithm for weapons jamming was altered to reflect the jam rate of real-world counterparts.

### **Version 1.7.0 Release –April 21, 2003**

Version 1.7 of the game was released April 21, 2003. Most of the development team was tied up with preparations for the Electronic Entertainment Expo (E3) in Los Angeles the following month. Since there was no time for a proper update, the only addition to the game was a new single map, a special-edition version of the popular Bridge Crossing. Although the team did not plan to increment the version number with this release, the Army requested it be labeled version 1.7. The internal version of the code was in heavy flux, so the previous version was rebuilt as 1.7.0. Unfortunately, a few bugs crept into the code packaged in this new version, while an improper assumption was made that the only change to the code was the version number; it was thus released without thorough testing. The result was a sub-par release that inflicted several critical bugs upon the community. Once again, the development team felt the pain of an inadequate testing solution.

### **Electronic Entertainment Expo – May 2003**

The E3 show in Los Angeles is about showing the world what new things are in store for your players. The tendency is to shove as much into the game as possible and somehow make it all work through smoke and mirrors. While the goal of the show *per se* can be met this way, afterwards developers find themselves with a roster of features and systems that are incomplete, in need of optimization and reworking. The *AA* development team spent the rest of the summer trying to deliver on promises made at the show.

### **Version 1.9.0 Release –August 8, 2003**

On August 8, 2003, version 1.9 was released, the biggest update yet. It was a difficult period for the development team, as there were more features needing

work than time to work. A comprehensive release would have to span multiple months, and the time jeopardized the schedule. It was eventually determined that this version was labeled as follows:

New Damage Model: The damage model for the game was changed for version 1.9. The damage, while the remaining health loss. If a combat medic resuscitated and further damaged, the concept of medics without a health bar was a dramatic change that

New Character Models: Before the initial launch of *American Assault*, the requirements for the game were never satisfied the team. A new set of characters with new, high-quality models, improvement, it entailed a complete

New Interface: The original interface was a minute, just before the initial launch for an ever-evolving product. The interface was unintuitive. For version 1.9, a new interface was thought put into navigability. The new website. While the result was an impression that *AA* was a very different game, incredibly tedious and time-consuming. The menu system, with such a vast amount of elements worked on it till, little by little, elements of the interface were completed over the next several

New Theme Song: Original theme song and augmented the new look. A new theme song was commissioned for the franchise. Nevertheless, it involved many changes and was finally approved.

Detail Textures: Capitalizing on the new artwork was created so that the game’s resolution texture was swapped



work than time to work on them. Although the plan had been to prepare a comprehensive release addressing all E3 fallout, it became apparent that the load would have to span multiple updates. The team's loss of two employees during this time jeopardized the schedule further. Examining the various features in progress, it was eventually determined that version 1.9 would focus on introducing medics: thus this version was labeled the "combat-medical pack." New features were as follows:

New Damage Model: To create combat medics for *America's Army*, a new damage model for the game was designed. In previous versions, all bullets inflicted a specified amount of damage on striking a player. This system was changed for version 1.9 so that the player initially suffered a percentage of damage, while the remaining portion was doled out over time in the form of blood loss. If a combat medic reached a wounded player in time, the bleeding could be staunched and further damage avoided. The system worked well by supporting the concept of medics without making it seem they had magical healing powers, but it was a dramatic change that players had to get accustomed to.

New Character Models: Because version 1.9 was released more than a year after the initial launch of *America's Army*, it was deemed acceptable to raise system requirements for the game. Most conspicuously, the game's character models had never satisfied the team. A decision was made to raise the bar and replace all characters with new, highly detailed versions. While the result was a dramatic improvement, it entailed a colossal amount of work for the artists.

New Interface: The original menu system for the game had been created at the last minute, just before the initial launch in July 2001, and its design was inadequate for an ever-evolving product. Aesthetically, it was unpleasing; operationally, unintuitive. For version 1.9, an entirely new interface was designed, with great thought put into navigability, expandability, and tie-ins to the game's official website. While the result was an extraordinary improvement that gave users the impression that *AA* was a whole new game, the work required to pull it off was incredibly tedious and time-consuming. There were so many pieces to the new menu system, with such a vast array of interdependencies, that the development team worked on it till, literally, the last minute. Of necessity, many smaller elements of the interface went unfinished, and polishing of the system would be completed over the next several releases.

New Theme Song: Originally, *America's Army* had no music. To open the game and augment the new look and feel, a distinctive, patriotic theme song was commissioned for the franchise. The development team did not create this work. Nevertheless, it involved many iterations and frustrating changes before the score was finally approved.

Detail Textures: Capitalizing on a previously unused feature of the engine, new artwork was created so that when a player got close to any surface, a high-resolution texture was swapped with the normal, lower resolution texture usually



seen from a distance. This allowed for a high degree of realism when studying world geometry up close, but kept system overhead manageable.

**Combat Medic:** To become a combat medic, players had to complete a four-level training sequence involving three classroom lectures and a field-training exercise. These levels were heavily scripted and presented actual first-aid techniques and quizzes. Much research went into making a realistic course, including consultation with medical professionals. Once qualified in the combat-medic course, players were able to treat injured comrades.

**Player Shadows:** Detailed player shadows were finally added to the game. This feature became available in the previous code merge and technology update, but required extensive engineering to work properly.

**Lip-Sync and Facial Animation:** In previous versions, facial expressions of characters were fixed. By licensing a middleware package developed for Unreal technology, the development team was able to add facial animations with speech synchronized to mouth movement. This capability, combined with the improved character models, boosted character realism tremendously.

**Punkbuster:** For a year, the development team tried to combat multiplayer cheating, but simply didn't have the time and expertise to squelch the growing number of hacks that were becoming available for *America's Army*. The job was finally contracted to a commercial anti-cheating firm, who added Punkbuster service to the game. The several weeks it took to port cheat protection to the Unreal technology were well worth it: the feature was a huge success with the player community, effectively stymieing those who wished to ruin the game for others.

**ROQ Video Support:** Support was added for ROQ-format video-clip playback within the game engine, expanding the team's ability to add supplemental content and offering another means of providing education about the Army.

**New Scoreboard, Team-Selection, and Class-Selection Interface:** In keeping with the new look and feel of the menu system, a new scoreboard and team- and class-selection interface was created. Unfortunately, there were so many elements involved with the new menu that it wasn't discovered till the last moment that we had failed to redesign these particular portions of it. Realizing the game could not be released without completing these elements, the development team spent the final days of the production cycle working feverishly to finish them.

**New Server Admin Commands:** An array of new commands was created so that those running their own servers could easily monitor, organize, and customize the game experience.

**Demo Recording:** We added a feature enabling players to record and view gameplay sequences within the game engine.

**Multiple Bug Fixes:** A great many longstanding bugs were finally addressed in this version.

## Version 2.0.0 Release - November 6, 2002

As a follow-up to version 1.0.0, the special-forces pack on November 6, 2002 had been originally planned for the previous release. This was the team's finest release ever. The

**"Special Forces" Role:** After players were qualified to play the special-forces (SF) role introduced in version 1.0.0, the ability to use and customize

**"Indigenous Forces" Role:** With the addition of training qualifications to play the indigenous forces role, this ensured that the missions were more challenging. A major point that a major duty of SF units is to operate in foreign countries.

**New Weapons:** The following

- SOPMOD M4 carbine
- SPR (SF special-purpose rifle)
- Thermite grenade (SF only)
- VSS Vintorez (enemy weapon)
- AKS-74U (enemy weapon)
- RPG-7 (enemy weapon)
- M9 pistol (snipers only)

**Weapon Modifications:** The ability to customize weapons by the player. A player could view their weapon and customize it to their desired. This was a major feature that added appealing aspects of play as a player. The following

- ACOG 4x scope
- ACOG reflex sight
- M68 Aimpoint sight
- M203A1 grenade launcher
- M583A1 flare launcher



**Version 2.0.0 Release - November 6, 2003**

As a follow-up to version 1.9, the development team released the 2.0 special-forces pack on November 6, 2003, completing another segment of the features that had been originally planned for that spring, as well as tying a number of loose ends from the previous release. Many players viewed Version 2.0 as the development team's finest release ever. The changes included:

**“Special Forces” Role:** After the successful completion of three training segments, players were qualified to play four new multiplayer missions as green berets. The special-forces (SF) role introduced new character models to the game, as well as the ability to use and customize an assortment of new weapons.

**“Indigenous Forces” Role:** We made it possible for players who did not pass SF training qualifications to play the new missions in the role of indigenous soldier. This ensured that the missions were available to all players while reinforcing the point that a major duty of SF units is to train and fight alongside indigenous forces in foreign countries.

**New Weapons:** The following new weapons were added:

- SOPMOD M4 carbine (SF weapon)
- SPR (SF special-purpose rifle)
- Thermite grenade (SF weapon)
- VSS Vintorez (enemy weapon)
- AKS-74U (enemy weapon)
- RPG-7 (enemy weapon)
- M9 pistol (snipers only)

**Weapon Modifications:** The SOPMOD M4 allowed a number of weapon customizations by the player. A new interface section was added by which players could view their weapon and add and remove interchangeable parts, configuring as desired. This was a major feature in version 2.0, and proved to be one of the most appealing aspects of play as an SF soldier. The customizable elements are as follows:

- ACOG 4x scope
- ACOG reflex sight
- M68 Aimpoint sight
- M203A1 grenade launcher
- M583A1 flare launcher



- Harris bipod
- M4QD suppressor
- Iron sight
- Heat shield

**3D Iron Sights:** Additional changes to weapons came in the form of true 3D iron sights. In previous versions, the iron sights for all weapons were depicted using 2D overlays. The new method involved three-dimensional geometry for more accurate portrayal.

**In-Game IRC Chat Client:** A new page was added to the interface to provide an in-game internet-relay chat (IRC) client, enabling players to speak with other users who were not necessarily playing at the time. This new tool further supported the community.

**Andromeda Server Browser:** Although for some time the game had employed licensed and proven server-browser technology, the Army contracted a third party to develop a new browser specifically for the game. In development several months, the product finally made it into the game in version 2.0. This technology never quite lived up to its design and proved a source of difficulty to the developers, and ultimately a major point of contention between the development team and the Army.

**Interface Modifications:** Continuing the work begun in version 1.9, the team made several adjustments to the new interface. These included:

- New progress bar for the server browser
- New mission-deployment page
- New in-game icon key
- New loading/connecting-message text boxes
- New glossary page
- Various detail settings on the video-options page
- Tour icons for the server browser
- Three new weapon-camouflage skins (desert, forest, arctic) were added to the weapon-modification page
- Resized server browser page (for better screen fit)
- Page and resolution sizing
- Ultimate Arena tournament server functionality for the server browser page

- An updated support

**New Weapon Animation S**  
feature, a new method was d  
animations.

**New Authentication System:**  
running the authentication s  
development of new authent  
was part of the game's tech  
required to make the transitio  
rough and there was an ext  
Additionally, it was not pos  
previous third-party compan  
information for an excessive  
frustrating aspect of this cha  
control of the developers, and  
decision to change, the burden

### Version 2.0.0a Release – 21 D

Originally unscheduled, this  
update over the Christmas holi  
then underway, the developers

**New Multiplayer SF Mission:**

**Resolved Punkbuster Issues:** S  
cheating system were addressed

**Distribution Partner and Versio**  
the Army to improve version tra

**Interface Adjustments:** Sever  
were addressed, including:

- Overlapping problems
- Unnecessary authentic
- Need for new authentic
- Changing the new-acco
- Changing the default in
- Updating game credits



- An updated support page

New Weapon Animation System: To accommodate the weapon-modification feature, a new method was developed for efficient display of third-person weapon animations.

New Authentication System: During this period, a third party took over the task of running the authentication system. Because of contract issues, this required the development of new authentication technology. Since the authentication system was part of the game's technological foundation, a vast amount of work was required to make the transition to the new company. Even so, the transition was rough and there was an extended period when authentication was unavailable. Additionally, it was not possible to transfer the full player database from the previous third-party company to the new provider. Because of this, account information for an excessive number of players was irretrievably lost. The most frustrating aspect of this changeover was that many elements were out of the control of the developers, and though the development team had not supported the decision to change, the burden of making it work fell on their shoulders.

### Version 2.0.0a Release – 21 December 2003

Originally unscheduled, this release reflected the Army's wish to provide an update over the Christmas holiday. Despite the detrimental impact on the schedule then underway, the developers effected the following changes:

New Multiplayer SF Mission: The mission "SF Sandstorm" was created.

Resolved Punkbuster Issues: Several operational issues with the Punkbuster anti-cheating system were addressed.

Distribution Partner and Version Tracking: A new system was created to enable the Army to improve version tracking and assess distribution efficiency.

Interface Adjustments: Several lingering issues with the game's new interface were addressed, including:

- Overlapping problems with the training menu
- Unnecessary authentication messages
- Need for new authentication messages
- Changing the new-account in-game URL
- Changing the default in-game IRC server
- Updating game credits



- Updating the support menu
- Adding server browser adjustments
- Adjusting the news page

#### Summation: March 8, 2004

Version 2.0.0a was the last release of *America's Army* developed by the MOVES Institute. In March 2004, the Army chose to take control of development and move the project off the Naval Postgraduate School campus. Although the MOVES Institute created one of the world's most popular video games for the US Army, differences between MOVES and Army management saw the game's production take a different turn. For many on the project, the whirlwind development cycle had taken an emotional and physical toll over the years. In the circumstances, a lesser team would have found it to impossible to deliver a game of such high caliber as *America's Army*, illustrating that the importance of selecting a team more for attitude and work ethic than seniority cannot be overstated.

#### LESSONS LEARNED

We obviously learned a lot from this endeavor, but three lessons are particularly salient:

- Pick the best team you can and support them. We accommodated our development team's creature comforts by supplying videogames and sofas for relaxation and (of vital importance) an industrial-strength, well-stocked canteen, and encouraged collaboration by offering a dim, cubicle-free workspace (allowing each to see what the other was working on and thereby to keep hold of the big picture). We assigned them a secretary for hated administrative chores and shielded them from direct contact with the client. Result: they stuck together and worked like madmen.
- Talk to your clients till you hammer out what they want, and have them sign off on it. If they choose to deviate, tell them in writing what alterations will cost in time, money, and the abandonment of agreed-on features.
- Don't just build a game; build the infrastructure for a game community. Our fan website proved of incalculable worth. Well beyond providing a forum for suggestions and bug reports, the *AA* site enabled far-flung individuals, alone at their computers, to become a tight-knit virtual brotherhood that circled the globe. The community displayed an intense regard for our development team; they were thrilled when a developer signed on to play, and the news spread like wildfire. The fans' pumped-up energy and

immediate approval  
inspiration through

#### CONCLUSION

We began this chapter with a discussion of combat-modeling systems for soldiers. We then showed how *America's Army's*. We discussed the problems and solutions in developing it. It represents a huge success, at the least one can expect. The difficulties developers face in creating a simulation in a government environment stresses and issues should be taken into account and heads stuffed with confidence into the game.

#### ACKNOWLEDGEMENTS

The authors salute the director of the school (YerbaBuena, 2004), for his support in playing online games—the school's (the MOVES Institute) or the school (the Naval Postgraduate School). We wish to acknowledge the producer of *America's Army*, Michael did a spectacular job. We are fondly of his time with the school, hiring, purchasing, and operating from 2000 to May 2004 is gratefully acknowledged as team mom kept admin from the flow of sugar snacks so near.

#### REFERENCES

- Davis, M., Shilling, R., Maier, M., Chang, C., Champlin, E., et al. *America's Army*. In B. Laurson (Ed.), (pp. 268-275). Boston: MIT Press.
- Macedonia, M., Zyda, M., et al. *Network Software Architecture*. Vol. 3, No. 4 pp.265-287.



immediate appropriation of the game was a source of refreshment and inspiration throughout our time on the project.

## CONCLUSION

We began this chapter under the premise that future training simulations and combat-modeling systems need to look and feel like games to be embraced by soldiers. We then showed how to organize a full game-development team, like *America's Army's*. We embarked on a history of *AA's* various releases and the problems and solutions involved. As an exercise in development, *America's Army* represents a huge success; we can look at the vexation level of its various setbacks as the least one can expect in such an undertaking, a lower bound on the difficulties developers can encounter. That going forward with game-based simulation in a governmental or corporate environment will always produce stresses and issues should be well understood. Nevertheless, with eyes wide open and heads stuffed with guidance, knowledge, and peer sympathy, let us stride confidently into the game-based future of training simulation.

## ACKNOWLEDGEMENTS

The authors salute the development team, pictured in the Yerba Buena guide (YerbaBuena, 2004), for their incredible efforts in producing one of the top-five played online games—the first game ever produced fully inside a research institute (the MOVES Institute) or based on a university campus (the Naval Postgraduate School). We wish to acknowledge Michael Capps as the original executive producer of *America's Army*, from May 2000 through the 1.0 release in July 2002. Michael did a spectacular job getting this project off the ground, and we think fondly of his time with the project. John Falby's role in making all contracting, hiring, purchasing, and operations happen flawlessly and expeditiously from May 2000 to May 2004 is gratefully acknowledged. Thanks to Rosemary Minns, who as team mom kept admin far away from the development team and guaranteed the flow of sugar snacks so necessary for the game's proper development.

## REFERENCES

- Davis, M., Shilling, R., Mayberry, A., McCree, J., Bossant, P., Dossett, S., Buhl, C., Chang, C., Champlin, E., Wiglesworth, T., & Zyda, M., (2003). Researching *America's Army*. in B. Laurel, (Ed.), Design Research: Methods and Perspectives, (pp. 268-275). Boston: MIT Press.
- Macedonia, M., Zyda, M., Pratt, D., Barham, P. & Zeswitz, S. (1994). NPSNET: A Network Software Architecture for Large Scale Virtual Environments, Presence, Vol. 3, No. 4 pp.265-287.



