Executive Summary

Given the importance of models and simulations in public policy making, and the need to improve their effectiveness, the governmental and non-governmental model and simulation building communities should be striving to explore and build on other existing model-building practices. Some of the most interesting work being done is within the interactive entertainment industry.

Every day game developers reach an audience of millions of people using increasingly state of the art hardware and programming. Not only is the game development community at the forefront of PC-based visualization, it is also a leading developer of applied artificial intelligence, overall interface design, persistent worlds, network interaction, and other building blocks needed for next-generation models and simulations.

The mission is simple - to create a better understanding of how commercial game and simulation developers, practices, and technology can be utilized by a wider field of organizations that build and apply models and simulations in the area of public policy.

This includes identifying and detailing specific steps organizations and game developers can take to blend game technology and approaches with proven model and simulation approaches to improve existing and future offerings.

Any casual observer who has seen someone interact with a computer or video game can easily understand how games can quickly captivate their audience. With their exciting visual and audio power, computer and video games take the competitive and fun nature of games to an entirely new level. Combining simulation, strategy, and the ability to play alone (if partners are not available) electronic gaming builds on basic instincts for competition, interaction, and imagination that are instinctive in so many people. By combining these elements with instructive materials, or wrapping important content in a gaming package, the hope is to utilize the strength of gaming to elevate learning and especially strategic learning among players. This has been the key goal of many non-entertainment applications of gaming technology and methods. Unlike many simulations and models, games are designed to be…

• Challenging • Entertaining • Educational • Played repeatedly • Multiplayer • Designed so no two games are perfectly alike •

The ability for game developers, technology, and industry practices to have a positive impact on models and simulations built for non-entertainment purposes is significant. It is already happening, despite little concerted effort by groups on either side of the aisle. The mere fact that there have already been several public-private partnerships to create games and game-like simulations on subject matter such as peacekeeping, healthcare, university management, and hazardous chemical cleanup shows the potential if partnerships could be more aggressively pursued.

Gaming is by no means a replacement for existing model and simulation building processes and practices but it has tangible advantages that ultimately could result in wider, more flexible, and more versatile products. To ignore these contributions will directly affect the ability for any simulation or model to reach its full potential.

The advantages are not just one way either, game developers could benefit greatly not only from a widening of the market for their talent but also by gaining access to a whole cadre of simulation and model building professionals and data collectors in other sectors.

By specifying the case for more involvement, and by detailing the specifics of how to create more cross-pollination, it is hoped that better modeling and simulation will occur. It will not only take more specific overtures to create such a working environment, but also more successful projects and integration of skills and personnel across the public and private sectors.

Ben Sawyer
Digitalmill, Inc.

Ben Sawyer is the cofounder of Digitalmill, where he is in charge of strategy, technology, and business development. Located in Portland, Maine, Digitalmill is a technology development firm with clients worldwide. It has worked on a wide variety of projects dealing with interactive game development, including support for The Sloan Foundation's *Virtual U* game project. The company has produced two books on game development, numerous articles about developing games, and several market research reports on the gaming industry. Currently Digitalmill is working on *Virtual U 2.0*, and consulting on other projects that integrate gaming, education, and training. Sawyer has authored or co-authored more than 10 computer trade books as well as numerous articles on a wide range of technology areas including e-commerce, interactive game development, software marketing, and computer graphics. Publications include *The Ultimate Game Developer's Sourcebook*, published in 1996 by Coriolis Group Books.

To find out more about the author, please visit: www.dmill.com
Contact the author: bsawyer@dmill.com

Use of this Module

To find out more about the author, please visit: www.dmill.com
Contact the author: bsawyer@dmill.com
Contact the Foresight & Governance Project at the Woodrow Wilson Center:
rejeskiDW@wwic.si.edu

Enhancing Simulations, Models and Their Impact Using Interactive Game Design and Development Practices and Technology

In today's public policy environment, computer simulations have become important modern-age tools used to affect the education, debate, and implementation process for a variety of initiatives. Whether they are developed with supercomputers in the national labs at Sandia or Los Alamos or using off-the-shelf statistical packages and spreadsheets, complex models and simulations are critical in helping scientists, policy makers, media, and others forecast, examine, and educate people concerning the potential outcomes and effects of public policy.

Given the importance of these models and simulations, it is critical to examine if they are being built as accurately and effectively as possible. Are they easy to access and do they truly visualize their subject matter?

Although some very accurate models, algorithms, and approaches exist, simulation construction and model building is as much an art form as it is a science. Given the importance of models and simulations, and the need to improve their accuracy and effectiveness, the governmental and non-governmental model and simulation building communities should be striving to explore and build on other existing model-building practices and systems. Some of the most interesting work being done, and therefore worthy of examination, is within the interactive entertainment industry.

Simulations in a Box

Some of today's best desktop-based simulations and models can be found in your local computer and video game store. Products such as the venerable SimCity series, Microsoft Flight Simulator, Railroad Tycoon, and NASCAR Racing might not be the most complex models, but there are important aspects of these game-based simulations that are worth examining and evangelizing. Game products are also getting better at taking advantage of faster machines and more advanced development methods and online products like Ultima Online, Everquest, and WWII Online represent the state-of-the-art in persistent online simulations in which the models of the game world are constantly available and evolving based on the interactions of the inhabitants which include non-user characters and user-led characters.
Many organizations are turning to games and game developers for help in improving the evaluation, prediction, monitoring, and educational processes surrounding their policy development. During the last decade, the armed forces have turned to computer gaming companies and products for direct use and development of training simulations, but they are not the only examples. In 1993, the Markle Foundation worked with SimCity developer Maxis to create SimHealth, a simulation of the U.S. healthcare system. In 2000, the Alfred P. Sloan Foundation released Virtual U, which is being used to train a new generation of higher-education administrators and managers and to research

university management economic strategies. Although training is the most natural outgrowth of real-world needs and computer gaming, there are other inspirations and technologies that the greater model and simulation building communities can take from the gaming community. Not only is the game development community at the forefront of PC-based visualization programming, it is also a leading developer of applied AI, overall interface design, persistent worlds, network interaction, and other needs for next-generation modeling and simulation programs.

The Importance of Game-Based Modeling

The best way to make the case for the importance of game-based modeling is that it has already been successfully put to use. Products such as SimHealth or Virtual U were developed through the cooperation of commercial computer game developers and public policy groups. Private businesses as well are looking increasingly at game-based simulations for effective training methods. And it is the philosophy of game developers that no matter how simple or complex the underlying model or simulation is, it needs to strive for suspension-of-disbelief, be holistic, balanced, and packaged in an accessible interface. This makes game-based simulation a user-friendly training and educational option.

As computers have become more powerful, developers have pursued with even more fervor the "suspension of disbelief" principle in which a game world is so believably presented that players become totally immersed in it. Furthermore, the market goals of game developers is to sell their games as wide an audience as possible. These are not necessarily the key principals that vigorously guide models produced outside of the commercial entertainment industry. This difference is an important one because when models and simulations create closed-loop systems, and are made easily accessible, the result is more discussion, more feedback, and potentially more significant impact. Additionally, game developers with unique approaches and perspectives can potentially improve the accuracy of models – especially ones based more on empirical methods vs. data-driven models.

Ease-of-use is particularly important if a model is being used to simulate or forecast derivative issues and fallout. For example, a model that allows people to simulate decisions made in an economic climate that has resulted from a steep climb in energy prices might be created so one can examine the reactions and strategies of different managers or constituencies. If that model is difficult to use or is poorly presented, the ability to conduct further derivative simulations and surveys with it is hampered. As an overriding principal, however, game designers and developers are especially familiar with these types of constraints and the need to satisfy a specific audience or audiences.

The basic criticism of game-based simulations and models is that absolute accuracy is sacrificed in the name of balance, fun, and accessibility. Furthermore, the need for them to run on hardware that is far less powerful than a state-of-the-art workstation means that game-based simulations have not had the same level of detail or horsepower available in other styles of simulation and model development. These criticisms are valid enough to have become accepted as fact and have stigmatized entertainment-born simulations as far less worthy than larger, spreadsheet-focused statistical systems or models built on heftier platforms with strict scientific principles applied. To some degree, the growth of desktop computing power and rapidly increasing complexity of games should debunk some of these prejudices.

Today's games are exponentially more powerful and sophisticated than those of just three or four years ago. Furthermore, many game designers are utilizing increasing computing power and their own energy to create far more complex, and scientifically sound models. So while most games are clearly not as accurate as many academically and scientifically produced models, it is a mistake to say that game technology and practices can’t be melded into such models and simulations in an effort to improve them in a variety of ways.

Existing -- if somewhat dated -- criticism coupled with poor communications between game developers and other model and simulation building entities has hindered a more widespread use of game-based skills in greater modeling and simulation circles. This includes the rapidly growing impact of game-based learning products that package complex system simulations into forms that can provide excellent training platforms for large audiences including managers and key constituencies. The commercial game and simulation community also has benefited by making the most of inexpensive, mass-market hardware and software systems, interface design, and more. The result is a blending of absolute accuracy and relative accuracy to deliver a model that, while potentially less predictive than other approaches, is more easily digested by the user. The ability to package complex systems in an understandable package will drive the use of interactive media and game-based training by constituencies of all sizes.

The mission is simple - to create a better understanding of how commercial games and simulation developers, practices, and technology can be better utilized by a wider field of organizations that build models and simulations. This includes identifying the issues and tradeoffs that may be involved, but also detailing specific steps organizations and game developers can take to blend game technology and approaches with proven model and simulation approaches that improve both rather than compromise existing offerings.

The Advantages of Game-Based Models and Simulations

The advantages of game-based models and simulations can be broken into three main groups:

Advantages of design that encourage wider and repeated use, and amplify learning opportunities and strategic thinking among users

Advantages of technology and approach that include utilizing off-the-shelf consumer hardware, high-end visuals and graphics, and intuitive interface design

A background in developing both non-fiction and fiction-based models with incomplete data, or empirically derived data

Design Advantages

Any casual observer who has seen someone's total attention captured by a computer or video game, let alone a good game of chess or cards, can easily understand how games can quickly captivate their audience. With their exciting visual and audio power, computer and video games take the competitive and fun nature of games to an entirely new level. Combining simulation, strategy, and the ability to play alone (as playing partners are not always available) electronic gaming builds on basic instincts for competition, interaction, and imagination that are instinctive in so many people. By combining these elements with instructive materials, or wrapping important content in a gaming package, the hope is to utilize the strength of gaming to elevate learning and especially strategic learning among players. This has been the key goal of many non-entertainment applications of gaming technology and methods.

Unlike the entire population of simulations and models, games are designed to be, or often are, inherently…

- Challenging.
- Entertaining.
- Educational.
- Played repeatedly.
- Multiplayer - a special skill unto itself.
- Designed so no two games are perfectly alike.
- Technology Advantages

One has to be careful to categorize the technology advantages game developers can bring to a particular simulation or model. This is because in many cases the best technologies game developers create are essentially shoe-fitting other cutting-edge ideas and programming practices into the lower-end hardware, which makes up the majority of PC and console gaming systems. That isn't to say game developers are not technology pioneers -- in areas like applied artificial intelligence or artificial behaviors, game developers have done a great deal of quality work. In terms of interface design and fast 3D engines, game developers have also pushed the frontier. As PCs become more powerful, the gap between what is pioneered on high-end systems and what is implemented on lower-end consumer platforms is shrinking.

Whether the advantage is from applied or pioneered technology advances, there are several key areas where developers can help existing models:

Interfaces

Since game software is intended to ship as a commercial product to a broad audience both geographically and demographically, critical interactive premises and elements are a crucial aspect of the design phase. Interface design is not an easy process, and indeed a key element that slows the very distribution impact of many models produced outside the game development industry is that they are completely lacking an interface. It can also be detrimental if the interface has an overall academic, scientific, or engineering-like interface that makes it much harder to comprehend, process, or otherwise modify. Quality game products rarely suffer from this problem even if the interface is quite a bit more complicated (e.g. a flying simulation, or a large system simulation), because intelligent designers have years of experience providing users an intuitive and productive way to enable their interaction with their games. If not then reviews and word-of-mouth will instantly kill a product's sales.

3D Graphics and Audio

Cutting-edge 3D graphics and positional audio are invading the gaming industry as new graphics cards and sound hardware enable developers to provide rich workstation/supercomputer level visualization to consumers. While many algorithms and methods for enabling 3D graphics were themselves developed over a generation ago, game developers have become the leading population of programmers who can implement high-end 3D-based graphics on consumer level hardware – a task that is one part matrix algebra and calculus, one part code optimization, and one part magic. This is especially true on PC hardware where technologies like Direct3D and OpenGL are the key methods for displaying graphics, and DirectSound and EAX are the key technologies for positional audio. The group most intimately familiar with these technologies: game developers.

Artificial Intelligence, Characters, and Opponents

In order to make some models and simulations come to life they very often need to possess believable characters, be it a wingman in a flight simulation, an upset local population, or an opposing force in a strategy game. Aside from seasoned AI researchers, game developers are at the forefront of creating believable friends and foes. From issues like making a character find an optimal route through a building to creating opponents that strategically utilize the resources they are given, game developers day-in and day-out are trying to create challenging opponents and believable in-game characters. This can often be some of the hardest work associated with designing a successful game or simulation. Not only is it hard to design but also is difficult to successfully program. If anything helps in this area of technology, it’s practical experience.

Online Multiplayer Gaming

When it comes to massive amounts of simultaneous online users game developers are truly at the cutting edge. Dealing with the backend management of servers, latency issues (the time it takes one piece of information to transmit between two points on the Internet) and the overall speed of dialup modems, game developers have been forced to ingeniously tweak network code to enable cutting-edge online games like Quake, Everquest and Ultima Online.

Massive multiplayer online gaming is expensive to undertake so most projects won’t move in that direction. But as more and more simulations and models find advantages in enabling online play and interaction between users, the most experienced pool of developers and programmers for such applications will come from a game development background.

Model Development Advantages

There is no doubt that game developers have a wealth of model building experience. However, the models developers build will defer from models and simulations built for other purposes than entertainment and mass-market appeal. Furthermore, many types of models and simulations built by game developers are fictional in nature. Whether it's a model of a medieval castle, an interstellar spaceship, or running your own Caribbean island dictatorship, many games are fanciful in their settings. These can be seen as weakness or strength depending on how you look at it – we of course see potential strengths.

Exploring the relationship between relative accuracy and absolute accuracy is one of the defining issues concerning application of game-based development techniques for model and simulation building. Many games, be they real-world or fantasy-world based, do not always build models focused on purely accurate premises. While many games do strive to use a wide and deep range of accurate algorithms and scientific principles (e.g. basic aerodynamic theory for flight simulations, or basic economic theory for a nation-building game) the nature of games tends to require a mixture of accurate and vetted formulas with those that are more empirical in nature to cover areas that have no pre-existing modeling algorithms to utilize.

This is also true in terms of the underlying data for many games. In numerous significant cases the data used is highly accurate (e.g. many sports games use extensive statistics captured by leagues and third-party databanks) but in other cases data is derived or modified to fit the premise or simplified interface of a game (e.g. SimCity uses cost data that isn't true of real cities to simplify the game and underlying math).

In place of absolute accuracy most games attempt to focus on providing just enough detail to simulate the feeling or the key aspects of any model. For example, while providing key supplies and fuel for tanks is a critical element of warfare, the sheer monotony of such operations is almost never simulated, in even some of the most detailed consumer war games. Instead the refueling needs of slow-moving and short-range tanks may be integrated into an overall reduced amount of movement for a tank unit. This may imply the impact of the refueling process without actually requiring a player to deal with it in such detail.

The end result is that games always strive to use just enough absolute accuracy to keep the game's overall logic cohesive and the relative trends in the game inherently balanced. For example, in SimCity the game simulates the relative nature of crime levels in relation to lower property values. Whether the crime model in this aspect is perfectly accurate, or as accurate as more sophisticated models of urban crime that have been built, the game preserves the key issue that there is a discernable relationship between the cause and growth of crime levels within lower-property value areas.

At its core the games industry is made up of many professionals who were influenced greatly by the notion of extensive and holistic world building, not from the fields of science but from the worlds of fiction and theater. Be it role-playing games like

Dungeon’s and Dragons, the entire world of middle-earth created by author J.R.R. Tolkein, or the visual worlds of George Lucas, game developers are inspired by the idea of immersing themselves in worlds both real and imagined. Coupled with their technical interest in simulations and programming this has spawned some of the most highly successful mass-market computer models ever created. This is in contrast to academic- and scientific-based motivations that dominate other model- and simulation-building communities. These communities are more rooted in the scientific method and principles -- to strive for complete understanding of a subject, prove a specific theory, or create an accurate prediction of existing systems. While the inspirations and guidelines for what constitutes a successfully produced simulation and model are somewhat different, it should be considered that neither has to be mutually exclusive -- both as a purpose and as a community.

Thus the advantage of game developers working on a model or simulation is not just in their technical ability to render them but also in their creative underpinnings. The creative force that drives most game developers is what makes them natural “out-of-box” thinkers, as well as excellent observers of the world around them. This sort of “world-building” motivation is especially useful for creating models of complex ecosystems.

The Key Advantage

Most models are built for small audiences, and their less-than-interactive nature results in an impact that is only widely felt when a news organization relays the results of a model to a larger audience. The report of the outcomes however is never too deep since very few people actually use and interact with the underlying math, science and other information contained in the simulation in the first place. Thus, the impact for many developed models is fairly limited, and even if widely reported, than not necessarily deeply understood.

By packaging (or repackaging) models covering critical issues, developers can attempt to use the widespread appeal of games to reach a wider audience. By applying all the right tenants of interactivity and game design to a model, its author can seek to provide a far better understanding of the relative factors and their interrelationships to the simulation’s users. If that impact is also education, then through gaming many sophisticated models and simulations can become extremely effective training tools as well.

Gaming is by no means a replacement to existing model and simulation building processes and practices but it has tangible advantages that ultimately could result in wider, more flexible, and more versatile, products. To ignore these contributions wholesale will directly affect the ability for any simulation or model to reach its full potential.

The Goal: Models as games and game-styled models

There are two distinct approaches to injecting game industry technology and practices into the wider field of models and simulations:

- Creating an actual game-oriented simulation or model.
- Improving a model or simulation by applying game development techniques and technology to it.

Each approach has specific goals and requirements.

Creating an actual game-oriented simulation or model

The most typical approach is to package a model or simulation as a game. This requires the model to be interactive and to provide the key elements of a game. In order to qualify as a game, a simulation or model must adhere to the following principles:

- The player must be able to tangibly affect the outcome of the game.
- There must be an overriding goal/challenge as well as sub-goals and challenges to the player with positive and negative outcomes based on their actions.
- It must require mental or physical skill.
- The outcome must be uncertain at the outset.
- It must require the player to develop strategies in order to win or succeed. Those strategies needn't be apparent at the outset; in fact the discovery element of gaming is one of its most important strengths.
- It must offer multiple paths to success. Linear games tend to take the form of puzzles, which, while useful and entertaining are primarily about figuring out a specific question and not necessarily about formulating strategies.
- Players must be able to ultimately overcome most obstacles in the game. Only under certain circumstances does it make sense to provide a game that isn't at some point "winnable."
- It must be interesting and fun (relevant to its audience) to inspire repeated play.
- It can also be educational in nature, especially in the context of simulations and models.

This approach is best suited for applying models to educational and training situations. Whether the goal is to educate a constituency or train staff and managers, using game-based models for learning is an excellent application and one that is being used more. Not only do these types of training applications help focus people on strategic thinking, game-based models can also be utilized for competitive training, team-based play, and for appealing to younger people who are familiar with interactive entertainment. The entertainment aspect of games can also lead to repeated usage and help encourage otherwise unmotivated or disinterested learners.

Improving a model or simulation by applying game development techniques and technology to it

Models and simulations needn't be actual games to benefit from some of the model-building principals and software development techniques used to construct computer games. The desired outcome here is to create an enhanced interactive model utilizing much of the technological prowess and interface skills of game developers. It is important that scientifically tested models are enhanced, not compromised, by the assistance of game designers, developers, and game-oriented technology. Specific skills and technology that can be applied include:

- Providing a completely graphical and intuitive interface that allows people to rapidly and easily interact with the simulation and model.
- Utilizing various artificial intelligence techniques to create realistic inhabitants in a model as well as reactions to parameters set in the model by its operator.
- Utilizing state-of-the-art graphics programming to maximize visual feedback and accuracy.
- Developing models that have empirical elements and imperfect data. Since many game worlds and models are fictionally based, this is a skill that is commonly possessed by game designers and developers.

If more models and simulations were packaged in as polished a manner as commercial entertainment software, the result would likely be increased visibility, use, and impact. For example, a model that is little more than a spreadsheet would not garner the same publicity as one that features an attractive visual aspect. This is especially true when better graphics can help bring a model alive. Another benefit of using interactive entertainment principles is the ability to produce models and simulations that are packaged and available for download on the Internet, resulting in inexpensive but widespread distribution and exposure, easy installation, and rapid initial use. On the PC platform, no single community of developers is more capable and available to produce state-of-the-art graphics and "quick to immerse" software than game developers.

Identifying and Overcoming Past Problems of Commercial Cooperation

In the 1990s there were a number of efforts made to pull together entertainment-oriented simulation and model developers and other parties, such as the military. Most of these efforts had limited success because of the structural and cultural differences between government/non-government organizations and the interactive entertainment industry. It is important to review and address these problems to improve the cooperation between game developers and designers.

In the 1996 paper "Opportunities for Collaboration Between Defense and Entertainment Research Communities from the Committee on Modeling and Simulation" a subcommittee of the National Research Council (http://www.nap.edu/html/modeling/) documented the key barriers to greater cross-industry collaboration. However, while offering a few recommendations to improve the landscape, the panel missed some points entirely due to its high-level composition. Some of the obstacles also identified have been positively affected by structural changes in the industry since 1996. The following is a point-by-point review of the major barriers presented in that paper and how, if applicable, these barriers have since been addressed.

1. Cost and profit structures

The council highlighted the discrepancy between the military model of payment (cost plus allowable profit) and the much higher profit margins of the game industry where the most popular titles can earn profits of more than 100 percent. While the Department of Defense model may be a bit different from other governmental and non-governmental organizations, there is no question that work outside the game industry will be contract-style work, with little or no upside other than the potential for further contract or maintenance work. The result is that successful developers and larger studios and publishers (i.e. public companies like Sega, Nintendo, Electronic Arts, and Activision) will have little or no interest in direct development work of this nature.

However, most game development studios and especially smaller development teams do not attain the levels of profit highlighted in this paper. The paper also fails to consider that with continued consolidation in the games industry, there will be more developers and skilled designers interested in work beyond that of the increasingly competitive commercial game market. Despite the lack of a lucrative upside, substantial contract work can be a welcome revenue base for many competent, independent development teams.

Agencies and organizations should better evangelize the substantial work that does exist outside of the commercial development scene and the potential revenues available. Developers also need to be educated about the type of work that exists and how to properly pursue it. Work should be more aggressively solicited from development shops and organizations should resist trying to hire only top-tier developers that have little economic incentive to do contract work. However, beyond the top 20-30 developers and publishers exists a very capable stable of development teams that should be eager to take reasonable contract work, especially if these opportunities can be used by the developers as research and development for future commercial projects.

2. Development Timeframes and Structure

A typical game project has a very different timeline than some public policy or government projects. While high-end projects can take two or three years to complete, most games are developed over a 12-18-month timeframe. Very little of this time is spent on traditional R&D priorities. Most game developers do the bulk of their R&D alongside basic development work; even if their R&D work is done separately it is usually measured in months not years. The result is that game developers have little appreciation for traditional research and development work.

The Council's report also highlighted the fact that many games have a short lifespan and that little effort is given to upgrading or improving them as they age. Both of these points remain valid, although perhaps less so than in 1996. In most cases these problems arise due to the nature of the entertainment market itself, not because of specific practices adhered to for other reasons. The game market has always brought out sequel products such as John Madden Football 1999, 2000, 2001. However, only recently have developed begun to respond more favorably to updating previously released products with a series of upgrades and extensions over their shelf life of the original product.

Valve Software's Half-Life is a great example, the core product has seen numerous free, and paid add-ons developed for it by Valve, and developers outside of Valve. This new practice is limited to PC products as the architecture of consoles makes the cost of upgrades prohibitive. Many top PC-oriented developers such as id Software, Valve, and Maxis all have released several upgrades to existing hit products. The existence of the Internet and high-speed lines has made this increasingly easy to do. Developers who have pushed this practice have benefited from the upgrades and, along with other developers, there is an emerging appreciation within the industry for the idea of continuous development on pre-existing titles. As more developers realize the benefits of frequent updates, extensions, and progression of their core titles, this will become standard practice. As a result, developers will provide integrated tools and technologies to further automate this process.

Many titles or projects that would be developed in cooperation with government and non-governmental organizations would be designed to have much longer life spans with more continuous development and maintenance than many commercial products have. In the game market, graphical capabilities have grown so rapidly during the last five years that titles driven by their visuals become quickly antiquated, thus driving frequent upgrades. But for many modeling and simulation projects, visual effects will not be as significant. The lesson is that this is less a cultural issue than an issue of contracts, budgets, planning, and markets served with little or no competing products. Unconstrained by market demands for newer products, the latest graphic advances, and sales competition, game developers are more than capable of delivering a product that is developed with longer-term horizons in mind.

Concerning R&D issues, developers are again constrained by the market and their budgets, resulting in little formal R&D by developers and publishers alike. R&D is usually built into an overall products' development and is rarely conducted as a standalone process. This means developers are truly focused on finishing the desired deliverable and will not get bogged down in various R&D that may not contribute directly to the project at hand. However, the industry as a whole doesn't necessarily investigate much beyond its immediate needs, resulting in innovations that are usually built on applications of pre-existing research in AI, graphics, sound, etc. than on groundbreaking new discoveries.

3. Intellectual Property Rights

Game developers are notoriously protective of intellectual property rights, especially software code. This is sometimes quite a problem for institutions that want to have full intellectual property control over contracted work. This is exacerbated by the growing interest in open or public source projects in which the source code for the software is made publicly available for others to read. This style of publishing is popular for structural or academic purposes. However, this creates a significant barrier that can stall projects before they even start. In some cases, certain intellectual property control is out of the question. -- such as the right to sublicense a technology to others without developer consent or rights to game characters or settings that have been part of a pre-existing.

To overcome this problem, all parties must be better educated on the issues concerning programming code control. More often than not the desire to control programming code exists to protect other business interests (e.g. access to make bug fixes or update the product in the event the programming team is unable to do so). Fostering cooperation and communication about these issues can help developers overcome the conventional fears that have arisen during years of working to protect the perceived value of code, game engines, and techniques.

At the same time, potential contractors must recognize that the developers they seek often have unique, valuable engines and practices. Contractors need to understand how to work around this issue to gain access to this intellectual property while protecting their desired outcomes.

In many cases, better understanding of the motives for intellectual property control and uniquely structured contracts can overcome intellectual property issues. However, often the trust, experience, patience, and creativity to work through these issues are not available in the early going of a project. Providing example contracts that work out common issues can help speed such negotiations. Educational groundwork with developers that dispels false beliefs and fears about intellectual property issues is also needed. In general, the intellectual property climate -- especially as it relates to software code and technologies – is becoming far more open and cooperative.

4. Interaction and Sharing Between Communities

The paper thoroughly highlighted the need for increased interaction between game developers, academic modeling and simulation interests, and non-game industry organizations. Suggestions included organizing stand-alone meetings, attending major game trade shows and conferences, and reaching out to various training and academic interests. Even now, this advice is sound and much of it has gone unheeded. There is little governmental and non-governmental public interest presence at the two major game industry conferences held each spring. There are no general clearinghouses of information to help vendors and organizations find each other, nor is there much being done to highlight various resources available to game developers who might want help tweaking their own models and datasets.

Since the report was published, the Internet has made it easier to publish and circulate documents between groups and made it easier to identify and find development teams. The open-source movement has seen a number of game engines and programs published with public source code. However, the existence of the Internet has also led to a great deal of person-to-person matchmaking. Overall interaction has been unorganized and stunted the potential for strong, higher-profile partnerships.

Current Issues to Resolve For More Exchange and Collaboration

Having made the case, and identified past problems there remain today six key issues to work through if game development practices and technology are going to increasingly mingle with non-game models and simulations more frequently and successfully.

- Educate those outside of the game industry about the skills and technologies the game industry and its developers have. This includes defining the key ways applications of game development technologies and practices outside of the entertainment industry can be beneficial
- Identify and overcome past problems concerning commercial cooperation
- Educate non-game development people and organizations about the process of developing commercial grade games so they can manage these projects successfully, blend game technology and practices correctly with their existing work, and retain the right talent to produce such outcomes
- Highlight some of the special needs for games that are developed for non-commercial non-entertainment user groups as well increase game developers access to resources that government, academic, and non-government groups that may help them produce more accurate and tested simulations
- Educate the game-development industry about how it can work with groups outside its normal operating environments
- Foster more cooperation between those inside and outside of the game industry through publications, conferences, and actual projects to create much better exchange of techniques, technologies, data, and overall expertise

Creating a New Era of Crossover Products and Cooperation

In order to create a new era of crossover products and cooperation between commercial game developers and the greater communities of simulation and model developers a new era of cooperation and efforts to build hybrid products needs to take place. As stated, some of this is happening already by groups motivated enough on their own to seek working together. However, with better understanding of how such projects can, or have been done, and the results they've had more can happen.

To help start this education process potential contractors need to understand the game development process, developers need a better idea of the changes to their traditional designs and processes that would be helpful, and everyone needs to understand how projects like this end up working.

Understanding the Game Development Process

One major burden on the growth of gaming in non-entertainment training and simulation situations is that many organizations have little experience with the game development process. This includes not only moving a project through the engineering process, but also making key design and structural decisions that turn actual simulations and models into playable games. Also, from the outset there is limited understanding of how to properly initiate and find a capable development team and how to work with them once they have been recruited. The process of designing game software is unique because, unlike many other forms of software development, it requires strong interactive skills not found in every software developer. This factor places even more importance on the recruiting process.

Many of the basics can be described in a short primer:

> A game project costs anywhere from $500,000-$10,000,000 to develop depending on its purpose, asset needs (artwork, sound), destination platforms (PCs and home videogame consoles), and overall complexity. Only upper-echelon products (budgets in excess of $1-$2.5 million) see extensive budgets usually required only for titles with a large amount of art, video and other multimedia assets. A typical PC project takes 12-24 months to develop and has a budget between $500,000-$2.5 million. Console projects typically cost more because development takes longer and requires specialized developers skilled in the peculiarities of console development.
>
> Most projects are funded by publishers while some are self-funded by the development studios themselves. Top publishers develop many of their titles "in-house" with internal teams but external teams are also utilized[u1].
>
> Depending on the structure of a project, developers are provided an advance for a title. There are two ways budgets for a project are typically approached: A publisher will commit to funding the monthly burn rate of the development team after agreeing how many months a project will take to complete, or a project is broken into specific segments where costs and funds are allocated for each segment. In many cases some combination of these approaches is taken. A team may agree upon a specific burn rate for its members' work while certain tasks such as in-game video, sound effects, or music might be priced out specifically. Since many projects will tend to slip behind[u2], or encounter engineering or design changes, every budget includes standby contingency funds as well as rules where a project might be cancelled.
> Development teams usually range from 5-15 people for a medium-size project with teams easily scaling up at various points to 30-40 people or more when factoring in all the various people (including documentation, producers, voice-talent, etc.) who may actually work some amount of time on a given project. Other than general overhead and some technology costs, the bulk of development costs essentially go to payroll. It is also worth noting that relative to some other software development projects, game projects tend to be developed by a tight-knit

group. Within a development team usually only a small core part of the team design the project, with one or two being completely responsible for the finite design elements of a project.

Programming is usually also handled by a similarly small group where a typical core team has 1-4 programmers. Larger projects may have as many as 8-10 programmers but the bulk of these will be assigned to subordinate aspects of a large program, such as development tools, country localization, etc. It is usually only the art staff that balloons the size of a overall staff, and subsequently the project's budget, since with art and other media assets time can be saved by increasing the staff available to complete those aspects of the project. It is also common to find key programmers involved in design duties, especially on smaller sized projects.
Projects typically are designed on paper as much as possible to start. If time and resources exist it is not uncommon for design documents to be quite lengthy and even storyboarded. For many simulation and model style projects, however, there tends to be a bit of both initial programming and designing done while developers test basic concepts and set up the general framework of a project. Once a design is fleshed out and a basic framework is in place, development tends to methodically pace along, even though the true final gameplay, and crucial gameplay balancing might not manifest themselves until late in a project. This is the nature of a model or simulation game product - almost the entire game needs to be complete before the best picture of the gameplay can be ascertained. This aspect of development can sometimes cause problems for organizations unfamiliar with the process because they may not see the full capability of a project until very late in the development process. Indeed a project might not look like it will fail [u3] or necessitate major changes, until a development team is more than half way through it. Thus the level of commitment by any group developing a game or a game-styled simulation needs to be very high.

During development a project is tracked closely. Usually a producer is assigned to oversee the project's progress. Schedule, budget, and other aspects are reviewed, and adjustments are constantly made. In extreme cases of poor development or slipped [u4]schedules projects can be cancelled. As a project nears completion extensive technical and gameplay testing is conducted to ensure bugs are eliminated or at least minimized, and that a project offers valued gameplay. As a game enters the beta-testing phase documentation is completed and edited and other aspects of marketing and promotion are finalized, culminating with final testing and development of a launched product. In the case of PC games, it is not uncommon to find upgrades and bug fixes (known as patches) to be provided after a product is launched. Post-launch developers also might release editing tools for users, strategy guides, and other derivative products for successful selling titles.

Finding, and Evaluating a Game Development Team

There are many capable model and simulation developers out there but don't make the mistake of assuming that seasoned model builders or programmers can become successful game designers. More often than not those not skilled or experienced with interactive software simulations do not have the ability to succeed at it. Thus, if your organization wants to pursue a game-based approach to a particular model or simulation it is wise to work with a seasoned designer and development team. There are many designers within the game industry who are available on a consultant basis.

Conversely, not all game designers and developers can create a model from scratch in areas where the do not have previous experience. In most cases the designer and development team's expertise will require a fair amount of education. To cut down on research work by a development team, it is important to have someone who has deep expertise on the subject matter (especially previous modeling and simulation development experience if possible) to work closely with the development team.

The ultimate combination is an experienced game designer, seasoned topical expert(s), and a strong development team with solid project management to see it all through.

Prior to fielding a development team, the first step an organization should take is to write a 2-3-page synopsis detailing the goal and purpose of the project. Furthermore, it is helpful to identify key data sources, modeling resources, and experts that the development team can review ahead of time. Table 1.1 provides both a list of key synopsis questions to answer, and a menu of resources to gather. Finally it is important to have some rough estimate of the overall budget that can be committed to a project. With those elements in hand it will be easier to recruit a development team and have them prepare proposals to evaluate whether the project as proposed is viable.

Table 1.1

Synopsis Outline
What is the purpose of the project?
What is the basic idea for the software?
What is the user's role; goal or main challenge?
Who are the main and ancillary target users for this product?
What platforms (Windows, Mac, Linux, etc.) does the product need to run on?
Is there a design that already exists to work from?
Are there experts in the subject matter that can be made available?
Are any of these experts familiar with building simulations, models, or games?
What data, pre-existing formulas, models, simulations, or other information resources can be provided to the designers/developers?
What is the general structure of the project?

Who will be the designer(s); Who will be the developer(s);Who will they work with and report to?, etc.
What is the desired completion date?
What is the available funding for the project?

With a good synopsis and some background resources in hand the best place to find development teams is by using a combination of industry resource listings and agencies. Most independent studios are not easily found in the phone book, and nearly all developers don't look for RFPs (although this could change in the years to come as more projects are launched), thus at least in the foreseeable future organizations will have to initiate the search. Gamasutra.com, the leading B2B community for the game development industry, as well as the IGDA (International Game Developers Association) provide the best resources to start a search. Gamasutra has a listing of agencies that lead a search for capable talent, and the site also offers its own list of contractors within the industry. While by no means exhaustive, the list does provide a strong starting point. When utilizing an agency or other third-party recruiters, be sure to look for agencies with a history of quality projects, especially with simulation and strategy titles. Due diligence is important as agencies are not as embedded, fully connected, or as reputable as they might be in other more historical, mainstream industries.

When potential development talent is found several steps will help create a successful evaluation process. These are outlined in Table 1.2. An evaluation should be coupled with a request (possibly funded) for a game design document, a technical document that describes how the design will be actually developed, a proposed schedule, and potentially (also funded) a prototype of the product. Prototypes could be a series of screen mockups, or a fully working component of the project. Prototyping (or demos in industry parlance) is a common publisher practice used to evaluate teams which don't have and extensive portfolio of previously developed products to evaluate, and is also used to ensure a project team can work cooperatively to achieve the vision initiated by an outsider.

Table 1.2: Game Development Talent Evaluation Tactics

Tactic	Purpose	Judgment Points
Review past projects	To evaluate if the development group has a track record of finishing products as well as the extent of its modeling and simulation talent.	It is important to work with teams that have at least 1-2 finished products, especially commercially published projects. Green teams can easily fail. Obvious pluses are teams who have developed core simulation and strategy titles instead of titles focusing on a more story, puzzle or arcade form.
Review each team member's	To evaluate the depth of talent of	This is especially important for greener teams. Many new development groups

resume	the studio.	are formed with talent possessing a deep previous experience.
Form an opinion of the team's cohesiveness	Many small development studios, especially greener teams can fall apart. Loss of key talent or team cohesion can be a particularly problematic risk.	Find teams, however small, with long operating histories. Look for strong management, and don't rule out interviewing the entire development team.
Look for strong cooperative skills	Game developers are similar to other forms of creative and artistic talent, and that means not all developers find it easy to work with others and render other specific visions.	Teams should be able to demonstrate via previous work, written proposals, interviews, and even prototypes that they have a cooperative mindset. Interviewing past publishers and contractors may be a necessity.
Evaluate organizational and schedule skills.	Finding teams that can deliver on schedule is tough – projects tend to slip [u5]easily in the game industry. More important is to find teams that are able to manage well. As projects slip, teams that continue to stay organized and focused will eventually deliver a finished project, and at a minimum slippage [u6]of schedule.	Talk with past publishers and references. Ask about how a project is managed, and require in a proposal that a developer not only provide a rough schedule, but a process by which they will provide to their employer tracking of the development's ups and downs.
Finances and current workload	Most development shops, even experienced ones, are usually small businesses with less than $3 million in yearly revenues, and many may be juggling other projects or efforts. Financial problems at such firms, as well as strains from other deadline projects. can adversely affect your project.	Teams should always explain what other projects they have and demonstrate they have or can bring aboard the talent needed to support multiple projects. Avoid teams with greater than two active projects unless their history and depth is extensive. Ensure that a team can remain together full time based on the budget of your project alone.

In general most development teams will be small in size and capitalization[u7]. The better teams will have a track record of working on and completing projects (large or small) and will easily demonstrate not only technical but relational and organizational aptitude. Coupled with a sound design plan, contract, and a payment schedule (based on achieved milestones), a well-evaluated team should provide minimal risk to finishing a project and meeting its agreed upon specification. In time, we expect a cadre of development teams to emerge that may begin to specialize in non-entertainment oriented simulation games and game models. Until then organizations need to rely on a past history of projects and experience that may provide some equivalent level of competency.

Working with a development team

With a development team on board, and a development plan in place, an organization still has much to do to ensure a project's success. Many of the responsibilities are the same tasks commercial publishers perform with their development teams. The biggest difference is that developers will especially need help working on topics and purposes that are not the staples of traditional game consumers. Thus in non-entertainment fields, developers will require more support and expert cooperation from the hiring organization.

Key tasks to manage once a development team is ready include[u8]:

Providing the key vision for the end product

Chief among an organization's responsibilities is providing from the outset a strong vision of the final project. While many game projects actually begin some of the programming work before every last design detail is completed on paper, projects falter fast if there is not an overall specification for the goal of the final product. Full agreement between the developer and the organization employing the developer is essential. In many game industry cases this is the developer's responsibility to provide, however in projects where existing models, policy ideas, or other goals are the focus the hiring organization takes much of the responsibility to provide the vision. Working in tandem with a designer might help, but ultimately the best projects will initially start from a strong idea provided to the developer from the organization.

Strong specification can be provided not only in a well-written outline, but also in the form of personnel that can work closely with designers to help shape the design and evaluate the project as it proceeds. The most important aspect to this style of cooperation is that the people working with the developer are able to quickly grasp basic gaming and interactive concepts, not to mention basic strategies and formulas for modeling various items in software form. In cases where this style of cooperation is difficult, third-party consultants and designers can sometimes offer the experience and ability to bridge cultures.

Providing data, research, and expertise on the subject matter

Organizations that aren't able or willing to provide the necessary manpower or make available other resources (i.e. data, research, etc.) should be wary or resigned to giving a developer a larger budget to handle this legwork.

Strong oversight, organization, and schedule evaluation

Aside from resources, research, and data, an organization also has to provide strong oversight and cooperation on the pre-determined schedule[u9]. This includes monthly development reviews (which become weekly as a project nears completion), and an overall cooperative environment even when development hits various snags.

Cooperation to finish projects and adjust schedule

It is important to keep in mind that while projects slip[u10], and that it is unacceptable for projects to slip too much, that it is difficult to establish a rock-solid schedule for game development. Aside from art and other media assets, no amount of funding or manpower will likely make programming and design progress any faster. Eight programmers on a project are not necessarily any faster than two, especially in game programming where programming styles and the resulting source code are unconventional. Development is an engineering process and all engineering processes, especially software development, hits snags and unintended delays.

Therefore organizations must provide proper oversight and management of a development team coupled with well-timed support and accommodation or risk a deteriorating relationship that can poison a project. Aside from strong oversight to help with organization and strong support, there is little an organization can do to encourage a project that hits delays to somehow magically right the ship. In general delays of 10-25% of an original project's planned timeline are not uncommon.

Strong Partnerships Needed

The purpose behind explaining the game development process, including hiring and managing game development teams, is so organizations have a clear understanding of the commitment they must bring to a project, and the points that identify a strong game development partner. In all contracting relationships a customer-client relationship exists. However, game-based partnerships with clients outside the gaming industry will often span generations, geographies, expertise levels, business approaches, and work cultures. This is exactly the sort of disruptive energy this paper is implicitly arguing for -- and will be a strength when managed correctly. However, this same disruption is also at the heart of why so few projects have been attempted, even given some of the great initial reasons for them to be undertaken. The solution is to understand that such cooperative efforts will be unique [u11]partnerships and therefore must be especially strong partnerships.

New Tactics for Games with a Different Purpose

Developers will play a vital role in order for game technology, development methods, and game motifs to become bigger factors in the greater governmental simulation and model building circles. This can also apply to training, education, and non-commercial markets. Developers need to create software that employs many game principals and techniques as well as feature ideas and practices that are different from games they would otherwise develop. This is because the nature of games becomes more application oriented when in the hands of people using them for items other than pure entertainment changes. Games must accommodate features such as being able to tailor scoring systems, outputting variables to printers and spreadsheets, and even become more malleable in order to be applied to many different uses, learning opportunities, and overall situations.

We have identified eight design changes and other issues to be considered:

1. Flexibility is key

One of the key aspects of games is that a designer has the ability to truly control the premise, the desired outcomes, the variables, and much more to shape what they feel is the perfect game. In fact it's the ability to balance many disparate elements that can provide for the optimum playability of a game title. With games meant for audiences outside the commercial market, there is a tendency to need more flexibility to change and recalibrate a game's initial play balance and other features. This is primarily because users of simulations, and models outside of the entertainment industry, may want to stress or skew [VGC12]them in many different ways. Thus we recommend that games developed for government, educational and other non-entertainment needs focus on providing as much flexibility to their designs and operations as possible. While it's important to ship what a developer feels is the optimum configuration of the product initially, the more a game can be modified, changed, and new outputs created the better. Fortunately, within the industry itself there is a rise in game engines and games that allow for extensive modification. If developers can carry that premise over to their core simulation and model building, the results will be welcomed.

2. Game goals, challenges, and scoring systems should be able to be changed by the user as much as possible.

While developers might feel they lose a lot of control over whatever bias they want to convey in a product, there is a need for the operator or trainer to change these outcomes and goals themselves in many learning opportunities and other simulation situations. Their goal is to fashion [VGC13]a specific situation they want to test, create, or train with. Furthermore the sheer variance of many types of systems people may simulate outside the entertainment world (e.g. an oil refinery, a refugee camp, a flood emergency) will require flexibility for trainers who might utilize them. Since many products outside the entertainment industry will be applied within a controlled training environment, designers and developers should create systems that defer to the trainers' needs and let them interpret the final outcomes and goals for the player. Traditional rigid scoring, scenario design and goal structures [VGC14]would become recommended scoring systems designs, while the product would allow operators a greater control to initiate play with minor to major changes in that scoring design.

3. Variables should be able to be saved for export, and printing should be supported where useful.

It is nearly unheard of for a commercial game to offer any sort of printed output or variable export function. However, most game players usually aren't handing in homework assignments (they're usually avoiding them), nor are they looking to graph and analyze the multitude of variables that the encountered during play. Outside the entertainment industry though these are normal occurrences and needs. Developers should provide strong printing capabilities (not just screen dumps), as well as allow the user to save variables in useful intervals or on-demand to a file that can be easily brought into Microsoft Excel® or other leading spreadsheets and statistical packages.

4. Public source or open-source publishing should be pursued

The ultimate peer-review in the entertainment industryfocuses on quality, capability, and playability of a game as reviewed by trade magazines, and the sales generated by the market. In models and simulations for other constituencies the peer-review is much more about the methods, assumptions, data, algorithms, and other techniques used to create the program. Without access to the source code that isn't as possible. With access to the source code other developers, scientists, academics, etc. can work to improve and extend the model. Game developers must recognize this and work to provide public or open-source distributions of products constructed for non-entertainment purposes. This includes well-commented code, as liberal a license agreement as possible, and support documents that help other programmers and researchers get up to speed with the source code. In cases where derivative commercial interests may exist, public source or private permission-based availability of source with a restrictive license is quite suitable. While open-source advocates will frown upon source availability that isn't pure open-source, that shouldn't discourage developers from releasing some grade of publicly available source code.

5. Platform requirements will lag the game market.

Game developers typically will write many titles with a focus on cutting-edge hardware capabilities. However, most models and simulations outside the core gaming market shouldn't be written to such a high-level system platform. Many users of simulations and models outside the entertainment industry will lag the top of the game market user base by as much as two or three years in system capability. Prime examples are such game technology staples as a sound card, DirectX, joysticks, and 3D graphics cards. Unless the game itself truly requires it, developers should aim for the lowest-end platform spec possible while still providing the overall quality and visuals that are feasible from talented interactive developers.

6. Installation, update and tech support systems must be extremely well planned.

Non-entertainment products must feature higher quality and extensive installation and support systems to ensure maximum use and adoption. Entertainment products for the most part sport much smoother installation processes than they did even two years ago. However, because of the need to install various operating system components in addition to the product, games usually have a more complex installation process. Expect to find many consumers who have never installed multimedia or game software to have systems devoid of important game technologies like joystick drivers, and especially up-to-date versions of DirectX, a key technology for implementing games on the Windows platform. The result is that an installation process needs to carefully evaluated and basic component of the package, or users need to be instructed how to install such critical components.

Another key component not usually encountered by game developers in their traditional markets will be the need to support network installation and operation. Many markets for game simulation and modeling projects will be in institutions that wish to install or allow the running of multiple instances of their software. Game developers should ensure, whenever possible that their titles:

- Can be installed over a network
- Can be run from a central file server by a local machine
- Are devoid of cumbersome piracy prevention processes such as necessitating the CD-ROM to be present in a machine in order to operate the product.[VGC15]

In terms of support, expect needs to be greater than traditional game audiences. This includes configuring such issues as sound, music, input devices, and graphics cards. Providing strong troubleshooting documentation with a product or via a Web site will be important.

Expecting that many titles developed for non-entertainment oriented uses will be offered at zero cost, developers should also anticipate widespread distribution via the Internet. This gives the opportunity to embed into the installation process electronic user registration and user surveys. Developers might also include an integrated update system for extensions and patches, as well as in-game links to online support and information systems such as a Web site.

7. Expanded, thorough and quality documentation including elements focused on user training, key evaluators, and expert users.[VGC16]

Most game manuals are as brief as possible without leaving out information critical to playing a game. Only rarely do games provide documentation akin to a standard PC application. Since game-based simulations and models are themselves more application focused than traditional games it is necessary that the accompanying documentation be likewise skewed. Furthermore, while no one appreciates spelling and proofing errors, these errors tend to be amplified in the academic and scientific circles. Developers may find themselves working with these people when they take on game-based simulation and modeling projects.

In addition to basic operating documentation for a simulation and a model, expanded documentation areas might include:

- A syllabus and teaching guide for trainers that may utilize the product in a classroom or other learning situation. This might optionally include a test bank, assignment ideas, and tutorial information.
- Complete background on the program's structure and systems for simulation and modeling.
- A strategy or expert playing guide that couples program tips with background information on the topic on which the simulation or model is based.
- A reviewers' guide that steps non-game savvy evaluators (e.g. press, agency heads, etc.) through typical playing session.

A formal bibliography that details data sources, empirical sources, reading, and further materials that were used in constructing the simulation and model.

In many cases it is expected that many users will be familiar with the subject matter of the program. However, in cases where a key element of the project is to educate users on the product's subject matter, considerable background and tutorial information on the topic is a major component of the manuals and in-game documentation.

8. Beta-testing will be a difficult process, but also a necessary quality issue.

Game developers are accustomed to a world where internal and external testing populations are extremely savvy, not only about their hardware but also about games and especially about works in progress. Support groups and steering committees outside of the core-entertainment industry will not be as understanding of works in progress. Beta-testing games will be a very foreign concept for them and even the slightest problems or errors may cause them to stop their evaluations at the very point developers expect them to be deepening their evaluation. For example, many academics will wonder why they are being asked to find major game-play bugs or miscalculations before common grammar and spelling mistakes are fixed. Other testers may not understand how to focus only on a subset of elements that you think are working and instead critique those that a developer pre-qualified as not working in the current build. Furthermore, unfamiliarity with gaming principles and playability problems will further hurt the ability to test.

To overcome these problems, developers should qualify beta-testers from a subject matter peer-group carefully and use a very small group as opposed to the large groups normally associated with top titles. Developers should prepare to spend more time orientating testers as to what testing entails, tutoring the uninitiated in the process. If possible testing should be done in the presence of a developer representative so help with support issues is available. The use of traditional game players and game testing labs, despite topics that might not be of interest to the average player, should be utilized. In many cases these traditional testing avenues, even devoid of topic expertise, will prove more useful.

Adapting Talent and Practices Creates the Best Success

Game developers can't assume that what is perfectly acceptable practice in their market is the same for all markets asking for their skills. Many games themselves could benefit from many of the above recommendations. Still, it's important to recognize the key and peculiar needs of what will be a different market than the commercial entertainment market. By paying particular attention to the expanded uses of training, the lower-end features of the wider PC installed base, and the flexibility needs of the greater model and simulation user, community game developers will make the most of their talents.

Case Study: Virtual U

In 1997, The Alfred P. Sloan Foundation in conjunction with William Massy, former CFO for Stanford University and president of The Jackson Hole Higher Education Group, gave birth to the idea to create a computer simulation of university management in game form. The idea was to capture, as much as possible, the breadth of decision-making needed to guide a major university. Furthermore, since a university has many component departments and responsibilities, the people involved believed a comprehensive simulation might help people understand the impact each component of a university -- from admissions to facilities to faculty -- had on each other.

The initial plan was to create "SimCity" for universities. With Dr. Massy’s experience as a major university administrator and a seasoned economist and model builder, the foundation believed it could deliver on a vision to create new ways to educate. By packaging it as a game with the plan of fulfilling various scenarios to keep key components of the university's finances in balance, it was hoped the simulation would not only be accurate but relatively entertaining. The goal was to deliver a novel approach to teaching modern theory about university budgeting, operations, and leadership.

The Development

With Dr. Massy driving the overall design and algorithm development, the foundation made $1 million available to sponsor the research, design, and development of the model. It was decided early in the process to use a seasoned development studio. Underscoring the importance of strong due diligence and the interview process, the first development team resulted in a false start due to poor development skills. The second development group, Enlight Software of Hong Kong, did prove successful. The team's strong experience with similar management strategy titles, including its well-reviewed Capitalism program, proved a good match. The key to the development partnership was that an experienced development team was married with a topic expert who further enhanced the process by bringing his own extensive programming background to the project.

Development first focused on the underlying engine and data set. The primary design was developed in tight coordination by Enlight's chief programmer and developer Trevor Chan, and Dr. Massy. The design input was enhanced by contributions from an ad-hoc steering committee of various university professionals and researchers at the University of Pennsylvania's Institute for Research in Higher Education. The Hong Kong-based Enlight staff handled all aspects of the actual game engineering, including programming, artwork, and sound. Coordination with the Dr. Massy and Sloan Foundation personnel consisted mostly of emails with occasional face-to-face meetings.

Development moved quickly at the outset, but poor planning and program development management, coupled with development team turnover within Enlight, severely hampered the schedule. Additional production management with game development knowledge came on board toward the final 20 percent of the project, but had this help been available at the beginning of the project scheduling issues might not have been such a problem. In addition, some issues concerning its design may have been better addressed. In the rush to get the project going, some decisions -- such as whether to make a version for the Macintosh, or what all the goals and challenges to the player would be -- hadn't been fully developed. The model itself was exquisitely developed but the game, which worked from that model, hadn't been as well specified. This underscores the need to have a development manager in place with an understanding of gaming and development to guide the process.

Packaging the Product

Developing the software is just the first aspect of any game or software development project. In order for a product to be successful it must be transformed into a comprehensive product and/or project. This includes it being packaged in some form, development of a distribution scheme, and in the case of many game projects used for education or training purposes, a training strategy developed in conjunction with any marketing strategy.

It was decided early in the development of Virtual U that it would be a product sold to targeted end-users. The goal was to create an initial investment by the user that would encourage them to use the product once they acquired it. Two products were created -- a low-cost game version and a higher-cost administrator version that sold for $130 – and pricing was determined by examining the common cost of graduate and doctoral textbooks. To ensure wider use and impact a free, playable demo version was available online.

For the packaged versions, two sets of manuals totaling nearly 200 pages of documentation were developed. Documentation was an important aspect of the product given its depth and breadth. A multimedia tutorial was also created to help users visually run through the game.

Distribution was via mail order and through the Internet. Users can visit the Virtual U Web site and find ordering information or the available demo for download. Anker Publishing, a book publisher targeting the higher-education professional market, was brought on as a co-marketer and fulfillment partner. Anker proved more useful than other potential partners because they were willing to be quite flexible in promoting and marketing a product of such a unique nature. It may not be apparent right away the best methods to reach the widest audience so it's important to have partners that can be flexible and quick to move to make adjustments to the marketing mix and distribution for such a unique product.

Release of Virtual U 1.0

VU 1.0 was released in the fall of 2000 with a good level of press and paid media support. It has sold nearly 1,000 copies and the demo has been downloaded more than 15,000 times. A sampling of purchasers indicates that three or four other people have on average used each copy sold. To date more than a dozen schools and programs are using Virtual U as an integral part of their higher-education administration curriculum. With a further Sloan grant to the University of Pennsylvania the project expects to expand the number of classrooms using the software to more than two dozen, representing as much as 10 percent of the overall number of such classes that exist in the United States.

What Went Right

While the first version of VU was far from perfect, as a couple subsequent patches to the software illustrate, its release signified a number of successes concerning the application of game and simulation design as well as the partnering of game developers with academic simulation veterans. The key highlights of this success include:

Strong team produced a strong simulation

The biggest asset to Virtual U's success was that the key people working on the project had a strong vision for what the model would accomplish and success hinged on game development processes and programming skills. A strong development team and a seasoned designer provided mutual support, and the project was also properly funded which facilitated the hiring of experienced developers to work on it. The result of this strong development team was a strong product. Third-party reviews corroborated this and fellow game developers recognized the product as one of the ten best independently produced and developed game titles of 2000 at the Independent Games Festival in San Jose, Calif. VU had a polished interface, excellent user documentation, and professional packaging that, while not directly contributing to the software itself, was a factor in its overall impact and significance.

Software sported several novel ideas

Within the software itself a number of ideas went very well. The budgeting system for VU is one of the most advanced seen in such a game. The economic underpinnings of VU are some of the development and design teams' best work. Furthermore the program's ability to simulate a wide variety of institutions -- from small rural to large urban campuses -- is also, while not perfect, a great initial success. The program is also successful at simulating the nature of large-scale organization management itself. Many of the decisions don't produce explicit reactions but instead initiate trends and behaviors that evolve toward a desired result by the manager. While not as immediately gratifying for players this is a much more realistic portrayal of large-organization management.

Distributed development team worked

Another successful point for VU was its distributed development team. While the team had its delays and organizational problems it still managed to work well together with key elements of the project team located in New York, Jackson Hole, Portland, Maine, Hong Kong and Lawrence, Kansas. Periodic face-to-face meetings helped, as did diligent use of reports and emails. It's important to remember that many game/simulation projects outside the entertainment industry will need to work in this same manner. VU shows that these projects can be as successful, if a bit longer to complete, as their entertainment counterparts.

At the grassroots level the great thing that happened with VU was that an initial community has begun to form around the product. This includes a group of higher-education professors who are using the product with their classes and developing assignments, syllabi, and more around the project. While the project has targeted these users, the first adopters have mostly come from people who had read or heard about the project on their own in various articles or from ads in educational trade publications, saw a demo at a conference, or came across it via the Internet.

Availability of source code

At first the idea of publishing the source code to VU was not even considered. When the idea came open for discussion, that fact hindered the initial push to publish the source code because the developers worried about both the intellectual property issue and how they might feel compelled to provide support the source code. The developers also believed they hadn't commented or structured the source code for public source use. Eventually these issues and fears were put to rest and the source code was published in the summer of 2001. To date over 200 people have downloaded the source code and helped the project fulfill a sub-goal of providing as much documentation of its model as possible.

What Went Wrong

The key problem for Virtual U was that schedule and organizational issues hampered its ability to finish strongly and on time. Schedule delays are impossible to blame on one factor, but overall the inexperience with the complexities associated with this type of project resulted in some organizational problems exacerbated by many of the specific factors. Those factors included:

Use of a development team that was stretched thin

Enlight was an extremely capable and experienced company, but as Virtual U dragged on other projects within the company began to compete and take time away from Virtual U. Furthermore, developer turnover within Enlight at key points of the process damaged team continuity and delayed the project several months as new developers assigned to the project had to get up to speed with the existing code.

This is a problem many projects may face because on many occasions only a few programmers will work on a specific project and the loss of any member during development will inevitably cause delays. To combat this it's important for the coding and development process to stay extremely organized which is a responsibility of all involved, not just the development team.

The model and simulation engine took too much precedence

The strongest facet of Virtual U is its underlying model. This model, however, took too much precedence in the project to the point where less than desirable game aspects for Virtual U were brought forth. Virtual U's scenarios were planned after simulations development had begun instead of earlier where simulation features could have been planned in advance to support the scenarios envisioned by the designers. Aside from specific scenario tasks, Virtual U only features one way to lose the game - bankruptcy. This severely reduces the challenges facing the user in the game, and thus reduces the overall playability.

There is no doubt that Virtual U is a game: it has challenges, the player controls the outcome, and it looks and feels like a standard computer game. However, the initial development planning missed the potential for VU to have a more coherent and complete game premise. Subsequently, with limited time left and budget left when these issues were raised there was only so much game goals and play that could make it in the final stage of development. Much like the garbage in - garbage out metaphor, the more game in the plan at the beginning of the process, the more game out at the end.

The complexity of the software created obstacles to wider use

Not every game is easy to use. While the argument that a game interface and motif makes it more enjoyable and easier to use than an inherently statistical simulation or model, no interface can hide the underlying complexity of the subject matter unless the overall simulation is watered down. Water down a simulation too much and it's overall realism, purpose and/or educational value may be irreparably harmed.

While it's hard to categorize it as something completely wrong with the end result of version 1.0, Virtual U suffers from a complexity problem. In order to begin instigating more pronounced and interesting policy moves, it takes a user as much as three or four hours to properly acclimate themselves to the simulation. Even with all of its documentation the game is more suited for one that moves at a slow pace, and has a steep learning curve.

Much of VU's complexity may have to do with its concept and content rather than any outright failure of design. If anything, the design has taken a very complex subject and made it infinitely more accessible to its users. However, by categorizing VU's overall complexity as a failure the design and development team feels it can do much better. This includes improving feedback mechanisms for the player, improving the model itself, and improving documentation. Furthermore the project has begun listing trainers among VU's more experienced user base to help those who need more personal one-on-one training with the product as well as providing paid one-on-one training by the project team itself.

Initial assumption of user mix for the product was not correct

VU's initial priority target users were presidents, provosts, and deans of United States-based universities. Unfortunately, this group is also among the busiest so sales to and impact on this group has been lower than expected. Conversely exposure and impact on students, lower-level administrators, faculty, and department chairs has been higher than expected. To adjust for this the project has focused its efforts on lower-level administrators in its direct mailings, as well as professors and students of education. To keep the goal of reaching out to presidents and other upper-level managers the project has tried to arrange for demonstrations and interactive sessions at various presidential conferences, workshops, and retreats.

The lesson here is that many times models and simulations are built with the hope of educating and impacting very high-level targets. Unfortunately the combination of limited computer access, skills, and available time usually hinders the ability for personal one-on-one exposure to the software. Even the novelty and ease-of-use of a game might not be enough to overcome these obstacles. Instead, personalized demonstrations are best, especially at times that have already been set aside for exposure to new ideas such as conferences and workshops. In some cases, getting that first important exposure has resulted in personal orders for themselves and staff.

The Future of VU

Thanks to its initial levels of success, and further generous support by The Sloan Foundation, Virtual U 2.0 is under way. With so much learned by the initial experience of

Virtual U, the ability to develop a second version of the product and the continuing evolution of the project would prove exponentially beneficial to its overall goals. Learning from the mistakes and strengths of the initial product, some of the adjustments to the upcoming second version include:

- Gaming features will be amplified. Players will have more challenges and more ways to lose the game.
- More training functionality is in the product. Players can output values to a spreadsheet for analysis and assignments, plus the ability to tailor the scoring system and other key weightings in the game as an instructor (or player) sees fit in order the achieve desired goals.
- Printing was greatly improved in a 1.3-released patch that will be included in the 2.0 version.
- The model pays better attention to distance learning and public university issues, two key constituency issues that weren't as well detailed in the initial project
- The ability to turn off randomness in the model - a key component of its game capability was added to assist those who wanted to more closely examine the effects of the model without random factors present.
- Better testing of the game has been organized from the ranks of experienced 1.0 users.

Virtual U 2.0 is scheduled for release in January 2002. Subsequent to this release is a grant that will enable VU 2.0 to be available in its complete form free for download from the Internet. This will widen VU's impact and let its more than 15,000 downloaders experience all of VU without requiring a purchase. With an improved model, an improved game premise, and a burgeoning user base, Virtual U 2.0 should continue to have impact on the education and practice of university management for years to come.

Virtual U is not the first combination of public policy simulation and gaming. It may not even be the penultimate version. However, it is one of the more successful, best-documented, and more high-profile projects of its type. As it shows, there are a lot of ups and downs in creating a hybrid product from a hybrid partnership, but the results can be quite innovative and worth the effort.

Bringing it all together

By specifying the case for more involvement, and by detailing the specifics of how to create more cross-pollination it is hoped that more will occur and that the benefits from better models and simulations will increase. It will take not only more specific overtures to create such a working environment, but also more successful projects and integration of skills and personnel.

The ability for game developers, technology, and industry practices to have a positive impact on models and simulations built for non-entertainment purposes is significant. It is already happening, despite little concerted effort by groups on either side of the aisle or third parties to more proactively push the players on to each other. The mere fact that there already have been several public-private partnerships to create games and game-like simulations on subject matter such as peacekeeping, health-care, university management, and hazardous chemical clean-up shows the potential if more aggressiveness, education, and experience in building the hybrid partnerships takes place.

The advantages are not just one way either, game developers could benefit greatly not only from a widening of their market for their talent but also by gaining access to a whole cadre of simulation and model building professionals, and data collectors.

Every day game developers reach an audience of millions of people using increasingly state of the art hardware and programming. By bringing together that strong end-user talent, with the problem solving and education cause of academia, government, science, and non-governmental organizations a new generation of models, simulations will be built that will represent the best of many different skills and purposes.