



ROBOTICARM
SOFTWARE

Note

This is a living document. To read the latest version, go to www.RoboticArmSoftware.com.

Build or Buy?

Finding the Right Tools Solution for Your Game Company

Dan Goodman, Founder & Director, Robotic Arm Software
www.RoboticArmSoftware.com

If you're a technical director, VP, or CEO of a game company, you've probably been faced with the choice of developing tools in-house or using off-the-shelf solutions. If you don't have your next engine yet, you could license one with tools already in place – but is that the right choice? And, if you've already developed your own engine, then you're locked into building all the tools to go with it – or are you? In this paper, we'll discuss the pros and cons of building versus licensing game development tools, and suggest a third approach that can give you the best of both.

How did we get here?

In the past few decades, game consoles have gone from primarily cartridge-based to CD-based to DVD-based (and now, with the PS3, Blu-Ray-based), and the available RAM on those systems has gone from just a few kilobytes to around half a gigabyte. As game systems expanded in available memory, storage space and processing power, developers utilized this space and power to develop bigger, more visually appealing games. Instead of rewriting every line of code for each new project, game engines were developed that could be used to create completely new games by just changing the data. Team sizes grew substantially to generate this data and get it into the

Hardware advances and consumer expectations have increased the data footprint of the average game exponentially.

Build or Buy? Finding the Right Tools Solution for Your Game Company
Copyright Robotic Arm Software, LLC 2008

Last Update: 10/18/08

www.RoboticArmSoftware.com

Better tools are needed to keep pace with the increased data requirements of today's games.

game. The “data wrangler” became a new position on many teams (usually a junior programmer or a particularly technical artist or designer), whose job it was to help the less technical staff with this process. This often involved exporting data from a commercial tool, editing (or creating from scratch) one or more text files with a text editor, running a batch process to build the data, and finally running the game and testing it out (and potentially going back and doing the whole process over again). This process can be lengthy, especially over many iterations, and many companies have developed their own internal tools to smooth the rough edges of their pipeline, empowering the ones generating the assets (artists and designers) to build, preview and test their changes without assistance. Still, the game industry as a whole has a long way to go before the tools we are using catch up to the technology of our gaming platforms. Let's look at the solutions currently available.

Off-The-Shelf Solutions

It's almost unheard of for companies to create their own modeling, animation and texturing tools, simply because what's out there already exceeds the needs of most developers. Many of the common art tools come from the film industry, and have become a standard among game companies, which means there is a lot of art talent out there already familiar with the software. Most organizations leverage the plug-in and scripting functionality these art tools provide to export art data into their own formats. Some companies even build design tools on top of their favorite 3D modeling package, and although leveraging existing technology seems an attractive alternative to building tools completely from scratch, many developers find that trying to turn an art tool into a design tool creates more hassle down the road with performance issues and unexpected behaviors.

Other off-the-shelf tool solutions are packaged with game engines or game engine components. Middleware companies have sprung up in the past few years covering areas like physics, animation, AI, and just about every aspect of game development imaginable. Cobbling these systems to your own game engine, however, is akin to piecing Frankenstein's monster together. It may come to life, but it will probably be a bit clunky.

While many of the commercially available game engines have a

Off-The-Shelf

Pros:

1. Running quickly
2. Proven tool set
3. Tools familiar to staff
4. Smaller team
5. Focused team

Cons:

1. Black-box technology
2. Integration issues
3. Performance issues
4. Support issues
5. Limited options
6. Licensing terms

relatively mature tool set, investing in such technologies ties your game development to engine features, supported platforms, and licensing terms of the provider. When considering these solutions it is important to evaluate the whole package, including the underlying engine technology.

Game engine development is a trade-off between performance and specialization – in platform as well as in genre. A racing game can be tuned for fast linear progression, a fighting game can be tuned for two characters inside a small arena, and an open world game can be tuned for random level streaming. A truly high performance engine needs to take into account the type of gameplay the engine needs to support. A first-person shooter engine, for example, may not be suitable for a racing game.

When it comes to hardware, developers often take the “lowest common denominator” approach. Features available to all platforms are included; others are less likely to be supported. Some may be available, but have such poor performance on one or more platforms, that using the feature in a cross platform game would be problematic. Finding the right engine tuned for the types of games and hardware your company is pursuing will be a monumental task.

If you are targeting the Sony and Microsoft consoles or the PC, there are quite a few options for off-the-shelf tools, both middleware and packaged with full-blown game engines. On the other hand, options may be very limited if you plan on targeting handheld platforms or the Wii.

In the short term, off-the-shelf solutions are very attractive because they get your game team up and running faster than developing your own technology. Unfortunately, most licensing terms cost money for every SKU of every project, meaning the more SKUs and projects you have, the more expensive development becomes.

Developing Tools In-House

Many game companies come to the conclusion that off-the-shelf solutions are inadequate for their long term goals, and opt to develop their engine (and tools) in-house.

One of the most attractive aspects of developing tool technology internally is that you own the technology, instead of licensing it.

Off-the-shelf tool solutions make sense if your game development process can fit into a box designed by someone else.

Developing In-House

Pros:

1. Developmental control
2. Needs-specific
3. Amortize costs
4. Build expertise
5. Build company value

Cons:

1. HR issues
2. Project conflicts

This has several advantages. First, the cost of company-owned technology can be amortized across multiple projects. This contrasts starkly with the licensing model, in which development gets more expensive with every project. Second, owning your own technology gives you the opportunity to license or sell that technology to other organizations, further reducing the overall cost and increasing the visibility of your company within the industry. Third, many publishers will perceive that a game company that owns its own engine and tools is technologically superior (and therefore more desirable to work with) to a company that licenses its technology from someone else. This perception attracts more contracts and builds value in your company.

Having control over development gives your team the ability to focus on features necessary for your game projects, and to create a tools pipeline that works for your specific needs. Developing your own engine and tools builds expertise among your team – no one knows the tools better than those who implemented them. That expertise comes in handy when debugging issues or adding new functionality, and having the tools developers in-house means that when problems occur or questions arise, they can be addressed quickly.

Building a tools team can be a difficult and time-consuming task, and not nearly so much as developing a full game engine and toolset. Since game developers make money from developing games, not tools, and tools engineers are not necessarily hot-swappable with gameplay engineers, it is extremely difficult to justify keeping a fully dedicated tools team through slow financial times, especially when the current toolset is in a workable state and game projects are in crunch-mode. But, layoffs mean losing the expertise that developing your own technology gives you, and some of the best tools engineers may be hesitant to work for a developer that has a history of downsizing in their tools and technology department. Another solution may be necessary to counteract the highs and lows of tools development demand.

Maintaining a staff of expert tools developers can be difficult when times inevitably get tough.

Contracting Tools Development

Contracting tools development means owning your own technology and having all of the advantages – amortizing costs, licensing revenue, developmental control, and increased

Build or Buy? Finding the Right Tools Solution for Your Game Company
Copyright Robotic Arm Software, LLC 2008

Last Update: 10/18/08

www.RoboticArmSoftware.com

Contracting Development

Pros:

1. Developmental control
2. Needs-specific
3. Amortize costs
4. Build company value
5. Smaller team
6. Focused team
7. Influx of ideas
8. Reduced dev time

Cons:

1. Communication issues

Contracting tools development increases your staffing flexibility and allows your internal team to concentrate on game development.

company value – without the project conflicts and HR issues. It frees your smaller internal team to concentrate on making games and working on engine features, while bringing an influx of new ideas into your organization. These “tools for hire” teams can work within the framework of your custom game engine and desired workflow.

A company dedicated to game development tools also brings a degree of expertise that can considerably reduce development time. Tools development specialists can spot potential flaws in design before they become serious bottlenecks to production. They can offer advice on what tools would increase your productivity the most, and how to balance power versus usability. They can also help you integrate off-the-shelf solutions like art packages and source control into your overall pipeline.

Even if you already have an internal tools group, you can offload work to an external team when the workload is particularly heavy. This gives your company the flexibility to expand and contract the tools development effort as needed, instead of maintaining a full staff through the peaks and valleys of tools development (or firing and rehiring for tools positions during those times). You are also able to schedule development more effectively, since you could have nearly limitless external tools development staff at your fingertips and the ability to develop tools in parallel with one another, instead of relying on your much smaller internal team to develop everything.

Some of your internal staff may be afflicted with the not invented here syndrome. Some “rock star” developers believe that no one else could possibly come up with better ideas or write better code than themselves. They may also believe that outsourcing tools development means giving complete control over to a third party, but this is not true. With the correct communication lines in place, and the right attitude on both sides, your team can work closely enough with the outsourcing team to not only give you control over development, but allow you to benefit from the expertise of the external team.

Luckily, tools development is probably the easiest engineering work to outsource. Outsourcing engine tasks (like graphics or physics, for example) is difficult because it requires integration with the rest of your engine (which is probably being worked on in parallel). Outsourcing gameplay, which requires constant

communication between design and engineering is even more difficult as the requirements of design tend to be in flux for most of a game's development. Many tools, on the other hand, come after an engine or gameplay feature is, for the most part, complete. You wouldn't, for instance, create an effects editing tool before you knew what particle system features and data formats were going to be supported in the engine. For most tools, given a certain input, the output is easily defined and verifiable.

Communication is probably one of the biggest issues facing game developers on a day to day basis. Many of the problems that arise during development are caused by miscommunication – between publisher and developer, management and staff, art and design, design and engineering, and so on. It is such an important issue, that the entire production department spends the bulk of its time handling this communication. It is the key to managing expectations on all sides, especially with contractors. Only when you both understand one another's needs, will you be satisfied with the end result. Good communication will foster a relationship you will be happy with and want to continue on future projects.

The external developer should open as many lines of communication to your team as possible to alleviate concerns about the current status of development. Email to the external team lead should be available as needed. Milestones should be scheduled regularly, giving your internal team a chance to review progress. The off-site team should be available on-site after milestone delivery to give face time to discuss what was delivered, what to expect next, and what conditions have changed that may affect the schedule. Conference calls should be scheduled between milestones to relay additional feedback. It may be a challenging process at first, but ultimately a rewarding one.

Look for a contractor that specializes in tools development for games. Game industry experience is key to understanding your unique needs. Overseas developers may save you money in the short term, but communications issues (like time zone and language problems) may cost more in the long run. Also look for companies with strong development philosophy, strong engineering practices, and flexibility in both. The external team needs to integrate with your internal team – and your practices – functioning as an extension of your internal development, and

Developing strong communication lines and having the right attitude are both key to successful outsourcing.

Find a contractor that fits your company's needs and specializes in game tools development.

not a completely independent entity. An ideal company would offer technical support beyond the initial contract. Being able to contact the developers directly through email or by phone will help your internal team gain valuable insight when a problem or question arises.

Our Team

We understand that tools development is one of the most important aspects of our industry. Having the best tools for the job enables technical and creative staff to create and test assets quickly and easily, allowing more time for development and tweaking. We believe that an expert team of game development tools specialists can deliver the tools that you need to make the best games in the shortest time possible. This is the corner stone on which Robotic Arm Software was founded.

We specialize in game development tools. In fact, that's all we do. Our process keeps your team and ours connected throughout the development cycle – and beyond. We can implement your tool design or work with you to design a solution for your needs. We develop all tools related to games – design, art, scripting, audio, UI, effects, cinema, and more. We can develop a single tool for your company or a suite of tools. We take on projects of all sizes, large or small. We will develop the tools you need so your team can concentrate on what they do best (and what makes money for your company) – developing games. Contact us today to find out how we can help you achieve your goals.

Contact Us

Email: biz@RoboticArmSoftware.com

Phone: (415)692-1472

About the Author

Dan Goodman spent over a decade as an engineer and lead in the game industry, developing tools, technology and gameplay for a dozen published titles before founding Robotic Arm Software. You can contact him about this paper or just to say hi at dan@RoboticArmSoftware.com.

Build or Buy? Finding the Right Tools Solution for Your Game Company
Copyright Robotic Arm Software, LLC 2008

Last Update: 10/18/08

www.RoboticArmSoftware.com