GAMASUTRA
The Art & Business of Making Games

PRINT

Postmortem: LucasLearning's Star Wars DroidWorks

By Jon Blossom,Collette Michaud

*Editor's note: This Postmortem appears in the August issue of Game Developer magazine.*In the fall of 1998, Lucas Learning emerged from its shell with the offering of its first educational software product, *Star Wars DroidWorks*. The game combines first-person shooter game technology with solid educational content to create something different: a thoughtful game that's actually fun and helps kids to learn within the game medium.

In *Star Wars DroidWorks*, you take on the role of a rebel spy disguised as a Jawa droid engineer, assigned the mission of learning the art of droid building. You use your skills in solving several physical puzzles, collecting clues that lead you along the path to a secret factory, where Jabba the Hutt has been making evil assassin droids for the Empire. To defeat Jabba, you have to engineer droids that roll, jump, walk, and run. In many cases, you have to build droids with special abilities — they have to move heavy objects, see in the dark, or perform some special task — and you have to explore and apply basic physical principles in order to infiltrate the factory and re-program the assassin droids.

DroidWorks met with rave reviews from family magazines, online educational sites, teachers, kids, parents, and even from many hard-core gaming sites. We did see a number of gamers scratching their heads and thinking "what's the point?" but for the most part, people seemed to love it both for its entertainment value and its educational value. Eventually, we won several awards including the first-ever award for children's interactive software from the British Academy of Film and Television Arts, the NewMedia Invision gold medal in the children's category, the NewMedia Invision award in the entertainment category (against "pure" entertainment titles including *Age of Empires, Unreal*, and *The X-Files*), and the 1998 Codie award in the young adult category.

As a company assembled to create a new kind of educational software, we knew we couldn't create just another action game. In addition to the unique challenges introduced by an educational bent, *DroidWorks* faced the same technical, artistic, and design hurtles as strictly entertainment computer games. We combined gaming styles as diverse as 3D puzzle games used in *Tomb Raider*, engineering and outfitting games like *Terra Nova*, adventure games like *Monkey Island*, and RPG games like *Diablo*, and we took ambitious steps forward to tackle them all at once.

What Went Right

Imagine the scene: a bunch of *Star Wars* fans with over-active social consciences, culled from the computer game industry and the world of education and curriculum development, are dropped into a room and told by George Lucas to "make software that is as educational as it is entertaining, make it better than anything else on the market, and don't spend a lot of money." They're given a blank slate and are released from the starting gate, ready to create products that grab kids and parents with *Star Wars*, hold them with great game play and production value, and satisfy them with intelligent, thought-provoking content. What could possibly be more right?!

1. Simplicity, Freedom and Vision

George's three-point mandate to the design team was brain-dead simple. His philosophy — and ours — was equally simple: when kids play, they are experimenting, and by experimenting, they are learning. George gave us the freedom to innovate, just as our products would give our users the freedom to experiment and learn. He directed us towards software products that would give players the freedom to build, explore, and learn from the experience, that would mimic Erector Sets and Legos. He asked us to allow kids to make mistakes, learn at their own pace, and direct their own learning in a fun and open environment, then he let us go.

Collette Michaud, the project leader, started dreaming up product ideas. She had often thought about a *Star Wars* game in which you build different types of droids, and she eventually simplified the concept into one sentence: give players the opportunity to build any kind of *Star Wars* droid and see it animate. To any computer-savvy fan of *Star Wars*, that sentence immediately conjured a complete vision, and as soon as anyone heard it, they said "of course." George had been pondering a similar idea, and he immediately approved it as the product to launch this new company.

Distilling an idea always renders it more potent, and all of the ideas behind *DroidWorks* had been distilled to their very core before we started building. This made it easy to communicate the idea quickly, to get people excited, and to align them with a central vision that drew on all the good memories they had of their own childhood toys. We could see from the very beginning how cool the game would be, and the ideas were so concentrated they never lost potency for the life of the product.

2. Kid Advisory Groups

Usually, we as computer game designers have the luxury of designing games for ourselves. We are the target audience and we are the judges, so if our game makes us say "wow," it's a good game. For many of us making the transition from adult games to kids games, we had to realize we were no longer the target audience — we couldn't just build something we thought was cool, we had to build something the kids would think was cool. So we enlisted the help of a Kid Advisory Group, and building that into our production process was very Right.

We gathered a group of about a dozen kids in the product's target age range to help critique and design the product while their input could still change the game's evolution. Involving kids in the design process helped us make important design decisions, gauge the level of educational appropriateness, and make sure the game remained fun as we moved through the development process. Seeing the kids react to our ideas, seeing them grabbing onto the game, energized and enlightened us month after month, and working with the Kid Advisory Groups proved to be one of the most rewarding and useful parts of the development process. Their responses, always blatantly honest, could threw us into tears of laughter or depression, and their visits always gave us something new to think about.

These were more than focus groups. Our kid advisors came back several times and grew emotionally attached to the project themselves as they saw some of their own ideas filtering through the game. They became part of the team, helping the rest of us step back from our adult lives and watch the project through the eyes of our audience. Regular consultation with KAGs from the early formation of an idea has become standard practice at Lucas Learning, and each project gets new groups about every six months, "retiring" one group in order to assemble another with a fresh perspective. The experience we had working with kids on *DroidWorks* made clear to us the importance of timely, appropriate, constant feedback.

3. Subject Matter Experts

We also consulted on design issues from the opposite end of the classroom by enlisting Subject Matter Experts, fondly called SMEs. The SMEs are adults with interest, expertise, and experience in the field of study at the core of our games, invited to help brainstorm, give feedback, and check in periodically during the course of the design process. Like the kids groups, subject matter expert groups quickly

became a standard part of the Lucas Learning product development process.

As game designers, artists, producers, and programmers, we spend most of our days working with computers, piecing together code and artwork, working out budgets and schedules, and keeping the project on track. To create a game, you don't need too much else—you make most of it up, possibly based on historical research. To create an educational product, you have to check every detail, make sure everything you're teaching is correct and well-presented, and, like a teacher, you have to know your audience. We can't possibly do this full-time, so we look to the SMEs for their expertise.

For *DroidWorks*, our original team of SMEs consisted of a variety of science and math teachers, curriculum experts, science museum coordinators, and engineers. We spent a day with them at Skywalker Ranch, tossing around ideas and playing with the *DroidWorks* concept. We continued to talk with them during development, but focused on two — one a middle school science teacher from the Sacramento area and one a retired science teacher now working at the Exploratorium in San Francisco. We continued to meet with them every other month until the end of the project to discuss the status of the game and the educational content we had incorporated into each of the missions. These two helped us immensely in designing the product, pointing out flaws in our physics, and helping us present things in kid-friendly ways.

4. LucasArts, *Jedi Knight*, and the Lucasfilm Family

Lucas Learning benefited immensely from their family tree. A small, brand-new educational software company would have a very difficult time starting up without a powerful license, and our ability to use the *Star Wars* universe was never in doubt. Furthermore, we enjoyed a special relationship as the sister company of LucasArts, and they made many of their resources available to help us jump start our own efforts. We contracted with their Sound, Voice, and Music departments, used their testers, and took advantage of their cutting-edge game technology, including their proprietary Smush video compressor and, most significantly, the *Jedi Knight* 3D engine.

When we first started adding details to the *DroidWorks* concept, interactive 3D graphics seemed an obvious fit, but we weren't sure we could do it. Time and budget constraints meant we didn't have very much time to devote to core technology, and with all we had planned, starting from scratch would mean cutting several key features we had hoped to include. When we realized we could use the *Jedi Knight* engine, then nearing its beta phase, the doors opened up for us. Not only would we have a real-time 3D engine, we would also have a custom-built level design tool (Leia) and an animation previewer/editor (ModelX), and we'd be able to find artists who had used both — in fact, two of our three level designers had worked on *Jedi Knight*. Better yet, we'd be able to grab the code immediately after it had gone through years of development and debugging.

Significant work had to be done to adapt the *Jedi* code, though. Pieces had to be ripped out, twisted around, and somehow welded into a new framework to turn a first-person shooter engine into a creativity tool. Then, a software funnel would have to be fitted to translate all the information from the workshop area of the game into a form the world simulation engine could handle. The final structure felt like a digital Frankenstein's monster, but the first time we saw a custom droid standing in an empty room in place of Kyle Katarn, the hero of *Jedi Knight*, we breathed a huge sigh of relief and thought "maybe this can work!"

DroidWorks would have been a very different project without the *Jedi* engine. We could have tried to build a new simulation engine, using some high-level 3D API such as OpenGL for rendering, but it could easily have ended up taking much longer. We could have created a new level design tool, or hacked something into an existing 3D modeling package, but we would have had to solve a whole host of other technical issues. We could have written our own scripting language and interpreter, prone to bugs. We could have thrown our hands up in the air and made *DroidWorks* in 2D… and what a different game it would have been!

5. The Violence Issue

Almost as soon as we settled on creating the game in an interactive 3D environment using technology from a first-person shooting game, our Director of Content, Jane Boston, spearheaded a debate that raged up until the very end of the project. *DroidWorks* exists in the *Star Wars* universe, a universe of explosions, guns, death, and violence. How much of that should we include in *DroidWorks*? What level of violence was acceptable? How could we create a story about saving the world from the evil Assassin Droids, in a game with "wars" in the title, without resorting to violence? And how do we even define what "violence" is?

Eventually, we decided to avoid violence at all costs, to bend the design in any way necessary to avoid a problem that would otherwise end in violence. There would be no gratuitous explosions, no laser guns, no death (just total loss of power or shields), and the Assassin Droids would have to get by with only their "shock" value — literally, since they disable you by touching you to drain your power. The gamers among us groaned, but everyone got behind the idea, and, remarkably, it turned out to be one of the better decisions we made.

Implementing the non-violence mandate often proved next to impossible as we drew trick after trick from our game design tool chests, throwing away idea after idea when we realized how much violence had become part of our everyday game vocabulary. As game designers, we're used to looking for the fun solution, the easy solution, so we resort to standard game vocabulary like bad guys hidden around corners or doors guarded by adversaries so difficult you avoid them until you've completed enough puzzles to have gained enough power. We had to throw away that vocabulary and try to find a new palette, and many times we banged our heads for days trying to find an interesting solution, wishing we could just hand over a laser gun or blow up a bad guy. Imposing the constraint forced us to get creative and pushed us into new territory.

We're very proud we created missions that require the player to think rather than react, and in the end, some of the kid advisors who had complained of not having violence or guns thanked us, saying they enjoyed it more. They described having the time to really use their brains and learn something while having fun, rather than shutting off their brains in order to react to violent situations. Needless to say, parents thanked us too.

What Went Wrong

Actually, very few things went unexpectedly wrong in the creation of *DroidWorks*. The team just moved together so well that, time after time, what seemed to be insurmountable roadblocks from a distance proved to be minor bumps when we actually approached them. Not that we didn't have our share of mind-numbing challenges that left us with our hands in the air! We had aimed so high that we couldn't possibly include everything we had hoped for in the beginning, and many features we wanted to include had to be cut (Where's the print button? And why can't you plug an arm into the head socket?)

1. Jedi Knight

Ironically, the choice to use the *Jedi Knight* engine rather than rolling our own 3D world simulation system proved to be one of the worst problems we had to face, even though it's also one of the best decisions we made. In retrospect, we go back and forth trying to decide whether it was the right thing to do. Could we have saved the time and energy we put into working around its limitations by creating a new engine from scratch that would address our needs directly? We'll never know.

In choosing the *Jedi* engine, we hoped to benefit from its physics simulator as much as from its rendering engine. We wanted our game to teach simple physics in an open-ended environment, and we hoped concepts of velocity, acceleration, and gravity, would just fall out of the simulator. How wrong could we have been! The level designers closely control the world of *Jedi Knight*, and the physics engine had been tuned to make maximum fun out of running characters and flying projectiles. We had hoped to teach interactive real-world physics, not cartoon game physics in which the main character could run 80 miles an hour, nothing bounced, and nothing could be pushed! We expected a fairly robust dynamics simulation but wound up with sphere-based collision detection in a system that often couldn't properly respond to the collision of two moving objects. The lack of rotational forces sure made it difficult to build levers and fulcrums.

By the time we discovered how difficult it was to create complex combinations of physical puzzles within the constraints of the existing system, we had no choice but to move forward with the game plan. We constantly had to redesign the missions to accommodate the physical quirks of the engine, ultimately scripting many key interactions where we had hoped the physics engine would "just work." With heavy scripting by our level designers and a few custom modifications to the engine, we managed to create missions that worked well within the engine, but unfortunately they fell somewhat short of our original design intentions.

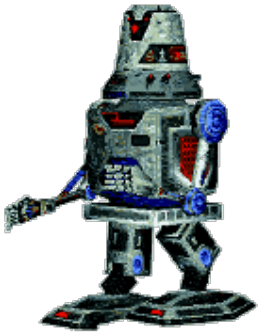
Designing for the *Jedi* engine posed other problems as well. In a game like *Jedi Knight*, almost all significant player interactions with the world has been scripted by the level designers and the animators, in a situation where they know everything about the character ahead of time. The character has been completely modeled and animated ahead of time, and his or her strengths and weaknesses have been determined well in advance. In fact, many physical constants are hard-coded into the engine itself! *DroidWorks*, we felt, must give the player choices during droid construction that have tangible physical consequences when they take their droid into the game world. "Make the droid matter," we chanted. In *Jedi Knight* or *Tomb Raider*, level designers know exactly how tall the character is, how far she can jump, how much she can lift, whether she can pick things up… In *DroidWorks*, we couldn't even promise the level designers that the character tackling their worlds would have legs!

Another hurdle we didn't see coming was the *Jedi Knight* art path. They had used 3D Studio to model and animate their characters, while our artists wanted to use Softimage. The *Grim Fandango* team had also chosen recently to use the *Jedi* engine for rendering, animation, and collision detection, and they had proven in concept an art path from Softimage. What the heck, we thought, it's just data…. After outfitting our artists with Softimage and starting the production process, we discovered hidden bugs in the data path that caused glitches in our animations and scaling problems in our models, problems that often required a complete rebuild for us to fix. Time after time, we found ourselves manually editing text files containing pitch/yaw/roll values to fix problems introduced during the translation from Softimage… Ouch!

2. Complexity: "Make The Droid Matter"

The number of combinations possible in *DroidWorks* gives it its power. How many droids can you build from 87 droid parts anyway? At one point during development, we could build over 65 million fully functioning droids, but design decisions forced us to cut it back to "only" 25 million by the time we shipped. How do you deal with that kind of complexity?

To begin with, we knew we could never test the game completely. It would take one person 290 days working round the clock building one droid a second just to build them all, let alone test them. A team of ten testers building a droid every minute in a non-stop 8-hour day could do it in 14.25 years if they agreed to work 7-day weeks. Like a sim game, we knew we would ship the game without complete testing coverage. Our testers accepted that challenge and did a great job covering a huge variety of droid types.



Our level designers also saw the nightmare of complexity. They could never be entirely sure what kind of droid would be walking into their rooms, and they had to leave the levels as open as possible. In many cases, they created ingenious physical filtering mechanisms that would guarantee certain types of droids beyond certain points in the level: a steep hill would weed out biped droids in favor of droids with tractor treads, a chasm would weed out wheeled droids who can't jump, a narrow canyon would weed out wide droids, and a short door would weed out tall droids. They used the terrain leading up to key puzzles to reduce the complexity of the puzzle itself, giving kids a chance to uncover the mission requirements themselves within the context of the game rather than being told "you can't bring that kind of droid here."

3. Positioning

In part because we used the *Jedi Knight* engine, and in part because we designed it for fun, *DroidWorks* looks more like an entertainment title than a traditional educational product. In our minds, the real beauty of *DroidWorks* lies exactly in that constant tension between entertainment and education. If we hoped to attract the attention of eight- to twelve-year-olds, we felt we had to give them something that could compete for their attention with the likes of *Quake* or *Jedi Knight*, something that would interest them and that they would have fun playing — a game! We believed we could combine traditional game-play elements with new and interesting kinds of physical puzzles, creating a game you think through rather than shoot through, and we pushed hard to steer the company in that direction.

From the beginning, our Kid Advisory Groups gave the game rave reviews, but many adults greeted it with confusion. How could something that felt so game-like be educational enough the be called a "Learning" title? How would we explain it to the public, who expects to find products on the "game" shelf or the "education" shelf? At some point, reality hit us full in the face, and we had to decide how to position the product and the company. Which shelf should be our home: entertainment or education?

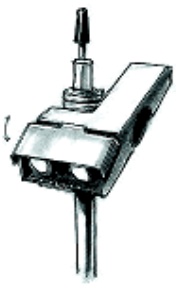
We had very little data, other than hunches. Who made the purchasing decisions in this age range? Parents typically look on the Education shelves, while teenagers and pre-teens typically look on the Game shelves. Teens wouldn't want to play a game they found next to Barney or Barbie, would they? Additionally, we were piggy-backing on the LucasArts' sales force in order to get into CompUSA, Best Buy, WalMart, Costco, and other big stores… and all their distribution channels led straight to the entertainment shelf. In the end, that's where *DroidWorks* landed, too.

Landing in the entertainment category proved problematic for *DroidWorks*. On those crowded shelves, *DroidWorks* disappears amongst action-oriented shoot-em-up games. Its bright blue box stands out from the dark tones of the adult games and catches the eye, and it feels strangely out of place. Furthermore, retailers have shorter attention spans for products on the entertainment shelves than they do for educational products: if a game doesn't disappear in the first two weeks, the stores send it back. Despite its great reviews and the press coverage we garnered in the month or two prior to its release, *DroidWorks* had a difficult first month of selling, and we started thinking our particular education/entertainment blend might be just too confusing for consumers. Luckily, stores gave it a break — thanks to the *Star Wars* name and the muscle of LucasArts' distribution — and by the end of the month, sales had improved and things were looking up.

4. Timing

DroidWorks also had its share of timing problems. We originally planned a fairly luxurious development cycle that had us landing on shelves in time for Christmas 1998. About half way through production, after looking at the numbers, Marketing decided we should aim for Labor Day instead, the official kick-off of the back-to-school buying season. Everyone agreed, so we took a deep breath, re-worked major pieces of the design, and made some heavy cuts. The team rallied behind the new date, and we managed to hit our sign-off date.

As the game took final shape, the company marketing effort began to kick into full gear. Until that time, Lucas Learning had no public presence. Our mission and message had never been projected to the outside world, so we weren't just marketing a product, we were marketing an entire company. We had lived in the cocoon of secrecy that typically surrounds George Lucas' endeavors, and the time had come to start opening that cocoon to let the world know what we were up to. That process turned out to be harder — and more time-consuming — than expected.



While the development team cranked away making the product, the marketing team cranked away designing the box, advertisements, and

collateral materials. The entire company watched in frustration as complete materials were proposed, created, and shot down in their final moments. Labor Day came and went, and we had no approved box. Magazine advertising deadlines came and went, and we watched in horror as the ever-present Christmas holiday approached. Finally, an approved package came down, and we landed on shelves in October, with our first magazine ads poised to hit newsstands in December, barely in time for Christmas.

5. Growing Pains

Although we had several aces in our hands, including guaranteed private funding and one of the most popular film licenses in the world, Lucas Learning was still a startup. We had to hire staff, find space, clarify our vision, define our culture, and adopt processes for doing everything. On *DroidWorks*, almost everything we did we did for the first time, and even though we had a lot of industry veterans on board, we often ran on hunches, not having time to pause the project to figure out the "right" way to do something.

Building a company while building a groundbreaking product is a difficult task. We tripped a few times, and we cobbled together many of our processes from bits and pieces of past experience. It turned out all right, and many of them worked well enough to be common practice in the company now. Others, however, fell apart when the light of day shone on them after the project, and there are many cases today where we look at what the *Droids* team did and say "this is exactly the wrong way to do this."

Ah well, we all learn from our mistakes… Centralize your asset database, even if you're working with outside contractors — we ended up using Access, FileMaker, and Excel to track various kinds of assets, depending on the format used by our suppliers. Don't try to track art production too closely, as collecting, processing, and keeping up-to-date all that information takes more time than it saves. Keep whatever assets require internationalization in a single database from the beginning, not scattered around in several obscure files that also contain non-internationalized assets. These are just a few of the things we did while we devoted our brains to other urgent tasks.

Success

However it sells or doesn't sell, we consider *DroidWorks* a huge success. We succeeded in creating a product that uses *Star Wars* in an engaging, non-violent way to teach basic physical concepts in an open-ended interactive environment that competes in production value and fun value with its entertainment-oriented peers. We succeeded, in the eyes of the press, in the eyes of players who have written us fan mail, and in the eyes of teachers who have developed lesson plans around the product, in creating a product with broad appeal. We succeeded in pulling together a great team of talented people, producing a quality game on time and on budget, and launching a brand new company, all at the same time.

DroidWorks
LucasLearning
San Rafael, Calif.
(415) 444-8800
http://www.lucaslearning.com

Release date: October 1998

Intended platform: Windows 95/98, Macintosh 7.5.5

Project length: 19 months

Team size: 15, including three programmers, four artists, and three level designers

Critical development hardware: Pentium 200MHz 60MB RAM

Critical development software: Form•Z, Softimage, 3D Studio Max, and Adobe

Photoshop

Project leader Collette Michaud and lead programmer Jon Blossom were two of the first employees at LucasLearning. They designed and built *Star Wars DroidWorks* with the help of an incredible team. Collette can be reached at collette@lucaslearning.com, Jon at blossom@aya.yale.edu.

Return to the full version of this article
Copyright © 2016 UBM Tech, All rights reserved