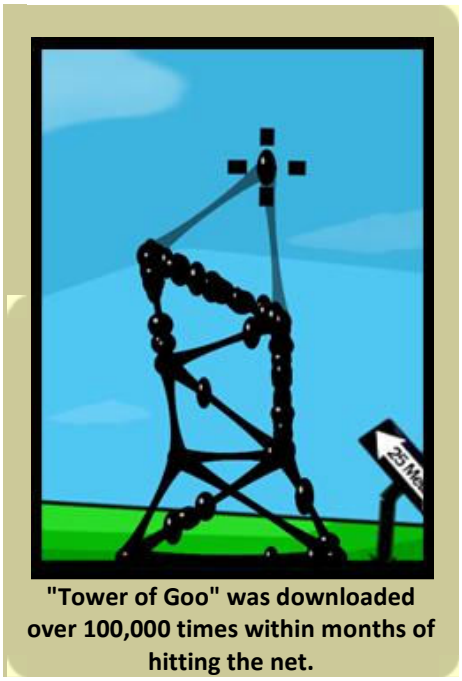
How to Prototype a Game in Under 7 Days:

Tips and Tricks from 4 Grad Students Who Made Over 50 Games in One Semester

By Kyle Gabler, Kyle Gray, Matt Kucic and Shalin Shodhan



"Tower of Goo" was downloaded over 100,000 times within months of hitting the net.

Here's a crazy game idea: Drag trash-talkin' gobs of goo to build a giant tower higher and higher. They squirm and giggle and climb upward over the backs of their brothers, but be careful! A constant battle against gravity, if you build a tower that's too unstable, it will all fall down.

"Tower of Goo" was downloaded over 100,000 times within months of hitting the net, it was dubbed “Internet Game of the Month” in one magazine, it was demoed on G4 and at the Experimental Gameplay Workshop at GDC, and it was one of over fifty games we made as a part of the Experimental Gameplay Project at Carnegie Mellon's Entertainment Technology Center.

And like the rest of them, it was made in under a week, by one person.

The project started in Spring 2005 with the goal of discovering and rapidly prototyping as many new forms of gameplay as possible. A team of four grad students, we locked ourselves in a room for a semester with three rules:

1. Each game must be made in less than seven days,
2. Each game must be made by exactly one person,
3. Each game must be based around a common theme i.e. "gravity", "vegetation", "swarms", etc.

As the project progressed, we were amazed and thrilled with the onslaught of web traffic, with the attention from gaming magazines, and with industry professionals and academics all asking the same questions, "How are you making these games so quickly?" and "How can we do it too?"

We lay it all out here. Through the following tips, tricks, and examples, we will discuss the methods that worked and those that didn't. We will show you how to slip into a rapid prototyping state of mind, how to set up an effective team, and where to start if you've thought about making something new, but weren't sure how. We hope these well-tested guidelines come in useful for you and your next project, big or small!

For easy browsing, all tips and tricks are organized into four sections: Setup, Design, Development, and General Gameplay. Enjoy!

1. Setup: Rapid is a State of Mind

Rapid prototyping is more than just a useful tool in pre-production – it can be a way of life! This section will show how to set up and begin thinking like a rapid prototyper.

Embrace the Possibility of Failure - it Encourages Creative Risk Taking

It's all about that little trouble-maker we call “risk”. Fear of failure, as far as we can tell, is the reason why movie licenses and double digit franchise games keep getting made. It's like always choosing to go to McDonalds instead of an unexplored new restaurant – always safer to rely on a well-known adequate option rather than take that risky plunge into the unknown but potentially delicious.

A good rapid prototyper would realize that failure is ok! That's what prototyping is for, so go crazy! If you fail, there will be dozens more, and chances are, you'll learn something anyway. By embracing the possibility of failure, rewarding experimentation becomes possible.

Mr. Gray: *“Mime After Mime” and “A Mime to Kill” were two games I made that attempted gameplay using only positional audio cues with no visuals. Although they were utter failures, the whole team was thrilled to take such a bold risk to prove the failure of audio-only gameplay, and I could point with pride to my hideous creations. As I gathered experience throughout the project, I was able to take more directed risks that lead to successful games.”*

"Mime After Mine" and "A Mime to Kill" - warmly embracing failure.

Enforce Short Development Cycles (More Time != More Quality)

You only need a few days. It seems like a natural and comforting thing to say, "Hey we made a great game in one week. Therefore, if we spend TWO weeks, it will be TWICE as good!" Of course this isn't the case. We found that generally any gameplay idea can be prototyped effectively in less than one week. Extra slop time tends to yield diminishing returns. Some prototypes, for instance, took just a single evening to throw together, while others got an extra week or two of love. Surprisingly, we found that there was no correlation between time spent in development and how successful the game ultimately turned out.

Attack of the Killer Swarm" (left) took just a day to throw together and surprisingly became one of the highest rated games of the project. "Suburban Brawl" (right) received an extra week of love but became so convoluted it probably would have been more fun without the addition of giant killer robots – which wasted both cities and time alike.

Constrain Creativity to Make You Want it Even More

Our most successful games grew out of specific themes or "toys", like "gravity" or "swarming" or "make a game targeted towards a predominantly female casual gamer demographic". Somehow, it became easier to be creative when there were restrictions in place.

Additionally, with a team of people all simultaneously generating prototypes around a particular theme, there was some guarantee we would avoid attacking the same obvious gameplay mechanics. Instead, we were challenged to explore and really suck the theme dry for all possible gameplay uses.

We moved away from this model towards the end of the project, ultimately to our detriment. Without solid thematic constraints, the games took longer to create, had less direction, and group unity deteriorated. There was less a feeling of "we're all in it together", and even worse, we lost the sense of friendly competition that was responsible for squeezing out those extra drops of creativity and finesse.

Some of the themes we explored were: "gravity", "springs", "evolution", "sound", "predator and prey", "addictive games", "drawing", "exponential growth", "vegetation", "balance", and a few others individually.

"Gravity Head" – use your gravity-powered head to grow and deliver.

Gather a Kickass Team and an Objective Advisor – Mindset is as Important as Talent

Each member of the team had to be comfortable with all aspects of game development. Everyone was responsible for their own programming, art, sound, and everything else that went into the final product. But talent wasn't everything. Ideally, it was important for everyone to approach this style of development with the understanding that design is paramount – everything else from art to engineering exists only to serve the final design. A great engineer without this mindset would likely be less successful than a mediocre engineer who fully understands this point.

A word from Jesse Schell, project advisor: *"I am always fascinated by the creative process for generating new game ideas, and so naturally I was thrilled when Shalin, Matt, and the Kyles proposed this project. I looked at it as an opportunity to try side-by-side controlled experiments in creativity, hoping there would be useful game design lessons learned. As faculty advisor, I tried to make sure that the team*

tried several different techniques, that the team was learning from their mistakes, that no one dwelled too long on an idea that wasn't working out, and that each individual was finding their optimal creative process.

"I gave suggestions along the way about how to improve the games as well, but mostly I tried to stay out of the way. I felt a bit like a gardener -- I did a little watering and weeding, but the flowering was all up to them. As this paper shows, the team was able to make some very useful conclusions – and ended up with some good games to boot! There is still more to learn about optimizing the creative process, and the Carnegie Mellon Entertainment Technology Center plans to continue this project."

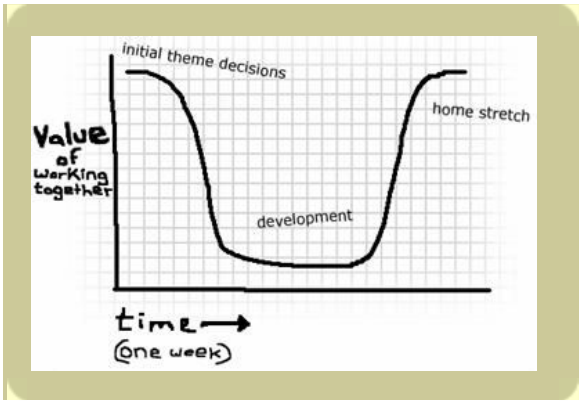
"Our team, Experimental Friends 4 Ever!

Develop in Parallel for Maximum Splatter

So once we assembled our team, what did we do? We stop working with each other! It might sound odd, but the benefits of not collaborating were too great to ignore:

- Risk Mitigation – By developing four prototypes simultaneously, we could make risky design decisions with the comfort that at least one or two were likely to be successful
- Friendly Competition – Everyone benefited from being kept on their toes. Like capitalism!
- Wider Thematic Exploration - Four minds all focused on the same theme forced us to plumb the depths of each topic. How embarrassing it would be if we all made the same game! This forced us into some rewarding creative realms and allowed us to avoid obvious points of attack.
- Sharing and Caring – Though we didn't share code (by choice, not requirement), we found it helpful to share concepts and understanding into a cumulative pool of knowledge. If one team member, for instance, discovered an effective way to represent spring systems, everyone would benefit.

As the weeks wore on, we found that having the team work together was most valuable at the beginnings and ends of each cycle. In the beginning of each cycle, the team was useful for helping to solidify and compare ideas. Once we hit development, we found each other to be more of a distraction than anything else – as each person was fully immersed in their own efforts. By the end of each cycle, we would all return to the room and work together until the wee hours of the morning for that last-minute extra taste of competition. A graph of this phenomenon might look something like this:



2. Design: Creativity and the Myth of Brainstorming

A great idea takes a split second to happen, but waiting for that lightning to strike can be excruciating. There's no such thing as forcing a great idea to squirt out, but this section should help cultivate your creative juices.

Formal Brainstorming Has a 0% Success Rate

We tried hard - boy we really wanted brainstorming to work! We scheduled "brainstorm meetings", and "powwows", we tried different color markers on whiteboards and oversized post-it notes, we even used motivational phrases like "blue sky" to help with our "out of the box thinking." But in the end, out of all the games we created, not a single one was the result of sitting down as a group for a brainstorm session.

Why not? This was all very shocking to us, but after much investigation, it appears that you just cannot schedule creativity. You cannot say, "Hey everybody let's meet for a brainstormer at 4:15, and by 5:00 we'll have 4 kick-ass game ideas ready to hit the ground running!"

But it's not hopeless. There are still (at least) two reasonable things you can expect from a well-conducted brainstorm session. The first, of course, is that it gets everybody thinking. And then sometime later, maybe on the drive home, or in the shower, or while taking Poopy for a walk, a brilliant idea will erupt in your head. Or maybe not. But as far as we can tell,

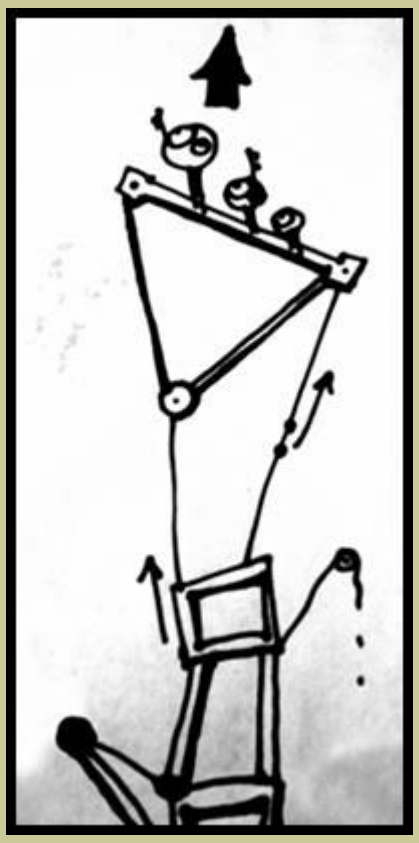
your mysterious brain does a lot of thinking when you least expect it.

The second way we found brainstorming to be useful was when there was something concrete to talk about. For example, “How can we improve this?” as opposed to “Hey let's come up with something arbitrary!” Given a half-formed game idea, for instance, it was moderately helpful to run it by the rest of the group to flesh it out. Everyone is a better critic than a creator, right?

Gather Concept Art and Music to Create an Emotional Target

As an alternative to brainstorming, we found that gathering art and music with some personal significance was particularly fruitful. People have commented that many of the games like “Gravity Head” or “On a Rainy Day” create a strong mood and have strong emotional appeal. It's no accident. In these and many other cases, the soundtrack and initial art created a combined feeling that drove much of the gameplay decisions, story, and final art.

Mr. Gabler: *“The idea behind “Tower of Goo” came up while I was listening to (for some reason) the opening to Astor Piazzolla's “Tango Apasionado” after walking home, and had this drizzly vision of a town at sunset where everyone was leaving their houses, carrying out chairs, tables, and anything they could to build a giant tower in the center of their city. I didn't know why exactly, but they wanted to climb up and up and up - but they weren't very good civil engineers so you had to help them. The final prototype ended up a little more cheery, and I replaced the final music with Piazzolla's more upbeat “Libertango”, but here's a case where an initial emotional target basically wrote the entire game.”*



Tango music and little men climbing higher and higher...

Simulate in Your Head – Pre-Prototype the Prototype

It's really easy! All you have to do is imagine your game audience saying, “Wow!” And then just work backward and fill in the blanks. What's making them enjoy your game? What emotion are they feeling? What is happening in the game to make them feel that way?

For each of our most successful games, it was never a surprise when they ended up being fun to play – in the best cases, we knew before touching a line of code that the idea was solid, because we had run a simulation of the game as a little thought experiment beforehand. The reverse is also true. There was no game that accidentally or unexpectedly became successful. We always knew ahead of time. (Unfortunately this didn't keep us from pursuing half-baked ideas.)

Simulating in your head also makes the development of the final prototype really easy. Since you will know exactly what you will be making, you won't waste time making expensive trial and error “design decisions” by noodling around in code.

One team member admits: *“It wasn't unusual to blow the first 3 or 4 days of the week just fooling around watching O-Zone*

music videos for "inspiration" or propped upside down in a bean bag listening to music and filling my head with blood occasionally running some crappy brain simulations. Finally Thursday or Friday would roll around and I'd panic because I still had no idea what to turn in for Monday, so I'd take the strongest of the ideas and tweak it based on whatever the obsession of the week was until it felt like a fun game, then stay awake for the next few days typing it all into a computer and drawing beautiful pictures. For me, (and I think all of us) the days spent in "pre-production" were unquestionably more valuable that the days spent in actual development."

An early paper prototype for "On a Rainy Day" - helpful while simulating in your head.

3. Development: Nobody Knows How You Made it, and Nobody Cares

Once you come up with a great idea, here are some tricks to whip up a little demo in no time!

Build the Toy First

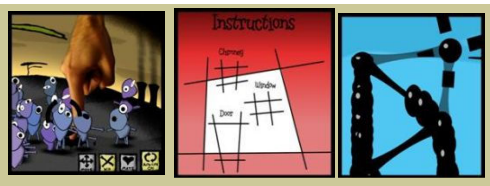
Start with the core mechanic. Whether spring systems, swarm behavior, gravity, etc, it never took more than a few hours to get the basic theme up and running. This "toy" should be the core mechanic of the game minus any goals or decisions. There is no win or lose state, just a fun thing to play with.

Mr. Gabler: *"With "Super Tummy Bubble", the "toy" was just a bunch of bubbles suspended in a little container. After playing with the toy and flinging bubbles around for a while, I got it tuned to a point where sticking my fingers in bubbles felt really satisfying, so it was time to slap on some gameplay. Gameplay features in this case were bubbles of different parasite types, a concept of "popping", a concept of "chains", score counters, etc."*

"Super Tummy Bubble" - Toy (left) vs. Final Prototype (right).

If You Can Get Away With it, Fake it

This is arguably one of the most important lessons of the project. Often the *"correct"* solution is not the *best* solution. Strategically faking it will save you time and money; it will make your game faster, and your teeth whiter. Fake it liberally and often! Don't set up complicated lighting and shadowing when a simple drop shadow and baked textures will be just as effective ("Darwin Hill"). Don't set up a complicated pattern recognition system for analyzing a user's drawings when you can fudge it with the same effect ("Suburban Brawl"). Don't draw splines or create your own vector art library when quickie stretched bitmaps give same effect faster and easier ("Tower of Goo"). This rule is also a fantastic general lesson for life, we have found. Slackers, take note



"Darwin Hill," "Suburban Brawl," "Tower of Goo" - all faked, and nobody noticed. Shhh!

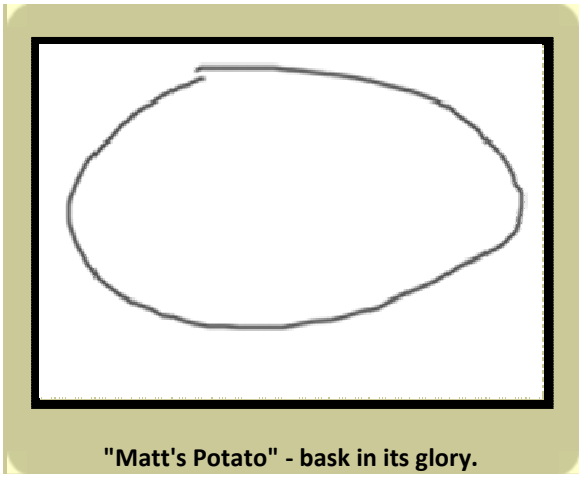
Cut Your Losses and "Learn When to Shoot Your Baby in the Crib"

At the beginning of the project there was a desire to salvage everything – a little more time and effort and surely a crappy game would become a work of genius! One such doomed prototype began as a beautiful spring system that squished and stretched and made you want to grab it and pull it all over the place, but it just wasn't become a compelling game.

The original spring system mechanic took just a few hours to create, but then it flared and consumed an additional wasted week of coding and re-coding in a sad attempt to force the mechanic into becoming a game.

It's important to quickly recognize dead-end game ideas, cut your losses and move on. As we found, spontaneity is more valuable than time spent trying to salvage existing code. You can always come back if lighting strikes at a later time.

Mr. Kucic: *"My "Potato" ended up being a perfectly simulated soft body system built in Flash. The only problem was that it was in no way fun. It caused more headaches than death metal, wasted a week, and it didn't even really move. You've got to know when to hold 'em and know when to fold 'em."*



"Matt's Potato" - bask in its glory.

Heavy Theming Will Not Salvage Bad Design (or "You Can't Polish a Turd")

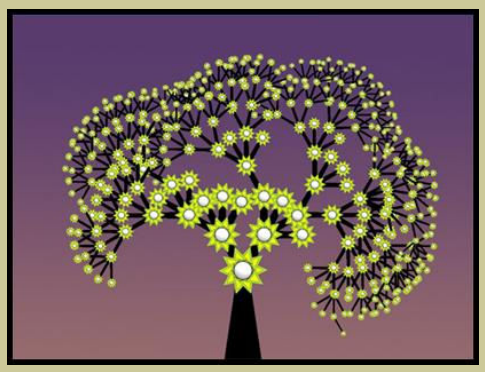
We found that game players are smarter than you think, and they can tell when you're pulling a fast one on them. If the gameplay is horrible, there is no recovery - all the art, music, and product tie-ins in the world won't make it a great game. Like taking a stale gameplay mechanic and slapping the latest 3D animated movie characters on it, nobody will be fooled.

Mr. Gray: *"With "Spin to Win", the 'gameplay' was to rotate your mouse to spin multiple circles – literally disc spinning. To hide the fact that it wasn't fun, I heavily themed the game with a '60s* Bewitched *art style and music. But no matter how much I polished the game, it still wouldn't shine. Despite all the extra love, it quickly became one of the site's most hated games."*

"Spin to Win" - all dressed up with nowhere to go.

But Overall Aesthetic Matters! Apply a Healthy Spread of Art, Sound, and Music

This is actually counter to one of our original hypotheses. We didn't think art or sound would make any difference at all, but we were wrong! Playing with a well-polished game actually feels better in your hands than playing the exact same code but with careless art and poor sound. It is important to make the following distinction though – polishing the aesthetic (as in the above section) will still not salvage bad design, but it does have the power to make a good game even more playable. This does not mean that you need fancy graphics or surround sound. It does mean that you can benefit from pulling everything together into a tight cohesive compositional package. Remember, even "crappy" can be a tight winning aesthetic if you frame it the right way.



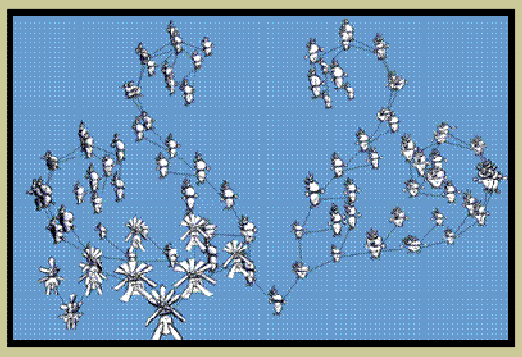
An unrealeased prototype - art doesn't have to be complicated to feel compositionally solid

Nobody Cares About Your Great Engineering

Again, it's worth noting that a great engineer does not necessarily make a great prototyper. "Correct" or "reusable" solutions are often not what we look for in quick throwaway code. For every problem, you should be able to come up with

a large handful of solutions and be prepared to pick the one that gets the job done – fast. The end user will never see your great engineering, and they don't care.

Mr. Shodhan: *"By over-engineering, it's easy to end up with generic tools or technology demos that never translate into something playable. This can be likened to a rock star executing a technically brilliant but entirely self-indulgent guitar solo that leaves the audience yawning! For the "evolution" round, I made a program with subdivision surfaces and a cell shaded look for evolving 3D models by cross breeding ancestry trees. There was a lot of cool technology, but it had absolutely no gameplay!"*



"Evolution Trees" - all that technology doesn't add up to fun.

4. General Gameplay: Sensual Lessons in Juicy Fun

In addition to learning the hard way about rapid prototyping, we also stumbled over some general gameplay guidelines. The following are a collection of some that significantly add to that "fun" experience.

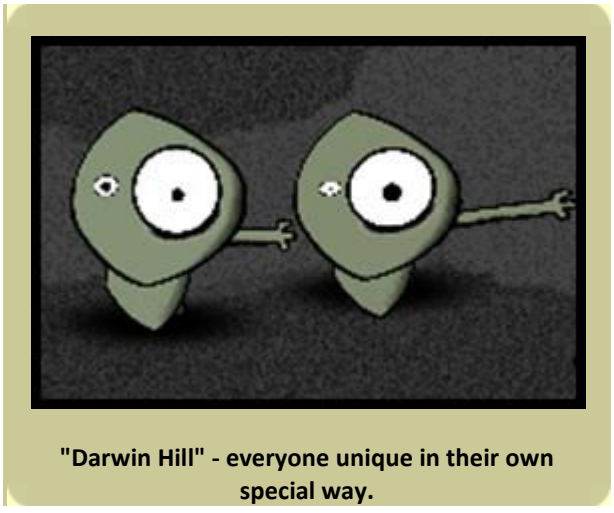
Complexity is Not Necessary for Fun

If mankind can entertain itself for literally thousands of years with variations on the theme of "ball and a flat surface", we might be trying too hard with some of this new-fangled video game stuff. It's entirely possible to have fun with basic primitives, think *Tetris*, *Pac-Man*, and any classic arcade game. Like the *Romeo and Juliet* love story archetype, these games had mechanics so good we're still reusing them again and again decades later. Lens flares, bump mapping, camera bloom, and other amazing new technologies are nice, but won't make your game more fun. Prove to yourself that your core mechanic is worthwhile with a simple prototype. Once you're convinced, *then* you can make it pretty.

A rubber ball. Watch out, 3D technology!

Create a Sense of Ownership to Keep 'em Crawling Back for More

We discovered quite accidentally that the games with the greatest replay value were the ones that had some sort of creation or customization aspect. For instance, "make a creepy tree out of hands and umbrellas", or "draw your own house", or "build your own tower", or "evolve your own race of mutated creatures". Apparently this is a well-known phenomenon and has something to do with all those create-a-face features common in many recent games, and customizable cell-phone ring tones, and those "be different, express yourself like everyone else" advertising campaigns. So jump on the bandwagon! Create a sense of ownership to keep 'em crawling back for more.



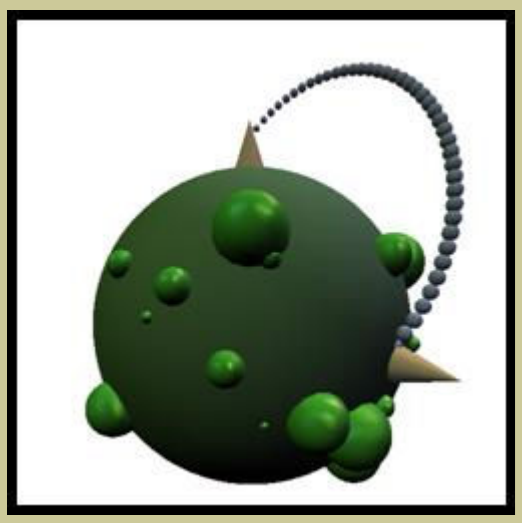
"Darwin Hill" - everyone unique in their own special way.

"Experimental" Does Not Mean "Complex"

Early in the project, many of the games we made were far more complicated then they should have been. Not only was the UI confusing, but the ways in which keys mapped to

actions were not natural or intuitive. Unless we could minimize the confusion time before the “oh I get it” moment, we knew we risked that players would get frustrated and never play the game again and possibly never come back to our site. Luckily, we found it was possible to be “experimental”, and still be easy to understand.

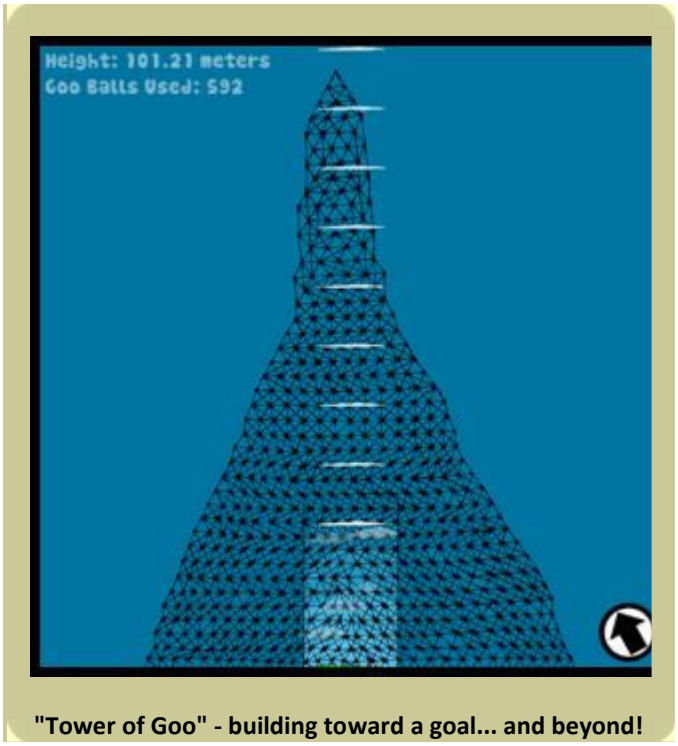
Mr. Shodhan: *“My first round game "Spaceball Munch" was a 3D version of the old "Gorillas" game where you specify an angle and a velocity at which one gorilla throws a banana at another. Not only were you now in 3D, but there was arc ball camera rotation and the game was on a spherical field. So there were two angles and a velocity to worry about. Oh, and it was no longer a discrete turn-based game but you controlled a continuous stream of particles and had to hit all these moving objects. Here is a screen shot of the gratuitous variable wastage.”*



"Spaceball Munch" - How can we know?!

Build Toward a Well Defined Goal

A well defined goal was embarrassingly easy to forget about. Without a gameplay goal, a prototype is just a toy – not a game. For some reason, people seem to enjoy having the opportunity to fail. A goal can be anything – like “collect x number widgets in a x amount of time,” or “keep the system stable,” or “traverse a space without touching anything bad,” but it was difficult finding a goal that didn't feel a little tacked on, (like anything involving a “time limit”). The best goals, we found, were an innate part of the gameplay like in "Tower of Goo," where the implicit goal was to simply “build up”.



"Tower of Goo" - building toward a goal... and beyond!

Make it Juicy!

“Juice” was our wet little term for constant and bountiful user feedback. A juicy game element will bounce and wiggle and squirt and make a little noise when you touch it. A juicy game feels alive and responds to everything you do – tons of cascading action and response for minimal user input. It makes the player feel powerful and in control of the world, and it coaches them through the rules of the game by constantly letting them know on a per-interaction basis how they are doing.

Some juicy examples you may have experienced might include:

- *Alien Hominid* – enemies exploding and flinging blood to an almost unjustified extent
- *Mario Bros.* – bouncing through a room full of coins, blinging with satisfaction
- Pachinko - a never-ending gush of balls all under your control
- *Super Puzzle Fighter II Turbo* – animation and sprites abound on multiple chains

Juice *feels* great! You can't keep your hands out of it.

Final Thoughts

The Experimental Gameplay Project team was a thrill to work with. We hope that the next time you try something new or a little crazy these tips and tricks come in useful for you. Who knows, that little idea you had this morning could be the next big thing. Pull some friends together, or go solo, but try and prototype it! You might surprise yourself.

Our objective advisor kindly pointed out, “Rapid prototyping can be a lot like conceiving a child. No one expects a winner every time, but you always walk away having learned something new, and it's usually a lot of fun!”

Happy prototyping!

This page reprinted from the Gamasutra Website:
http://www.gamasutra.com/features/20051026/gabler_01.shtml

Handy Cut-Out List!

Setup: Rapid is a State of Mind

- Embrace the Possibility of Failure - it Encourages Creative Risk Taking
- Enforce Short Development Cycles (More Time != More Quality)
- Constrain Creativity to Make You Want it Even More
- Gather a Kickass Team and an Objective Advisor – Mindset is as Important as Talent
- Develop in Parallel for Maximum Splatter

Design: Creativity and the Myth of Brainstorming

- Formal Brainstorming Has a 0% Success Rate
- Gather Concept Art and Music to Create an Emotional Target
- Simulate in Your Head – Pre-Prototype the Prototype

Development: Nobody Knows How You Made it, and Nobody Cares

- Build the Toy First
- If You Can Get Away With it, Fake it
- Cut Your Losses and "Learn When to Shoot Your Baby in the Crib"
- Heavy Theming Will Not Salvage Bad Design (or "You Can't Polish a Turd")
- But Overall Aesthetic Matters! Apply a Healthy Spread of Art, Sound, and Music
- Nobody Cares About Your Great Engineering

General Gameplay: Sensual Lessons in Juicy Fun

- Complexity is Not Necessary for Fun
- Create a Sense of Ownership to Keep 'em Crawling Back for More
- "Experimental" Does Not Mean "Complex"
- Build Toward a Well Defined Goal
- Make it Juicy!