Gamification by Design

Gamification by Design

Gabe Zichermann and Christopher Cunningham

O'REILLY®
Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

Gamification by Design
by Gabe Zichermann and Christopher Cunningham

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Printing History:

ISBN: 978-1-449-39767-8
1303155502

Table of Contents

Preface

The rise of gamification and the increasing power of game thinking create substantial opportunity for web and mobile designers, developers and producers. Major brands like Nike and United as well as startups like Foursquare and Groupon have used game mechanics such as points, badges, levels, challenges, leaderboards and rewards to create and engage community while igniting viral growth.

Implementing gamification in a new or existing website, however, can be complicated; there are hundreds of technical and process options to choose from, and the landscape is constantly shifting. *Gamification in Action* is designed to decode the possibilities and provide a series of proven and tangible strategies and tactics that any developer, producer or product manager can use to implement game thinking in their website.

Written by the foremost expert on the subject of gamification, Gabe Zichermann, and a key thought leader in mobile architecture, Christopher Cunningham, *Gamification in Action* is the first book to delve into the specific technical implementation requirements of a gamification solution. Practitioners at any intermediate or advanced level of skill should be able to implement Game Mechanics' specific tactics without hesitation.

Every web producer, developer and entrepreneur wants to deliver a sticky, viral and satisfying experience to his or her customer. Gamification enables web experiences that influence user behavior in astonishing and unparalleled new ways. This book shows how to integrate gamification into any kind of consumer-facing website, providing both strategy and tactics for leveraging this powerful technology.

Introduction

It's summer dusk. A neighborhood of children runs between trees and fireflies shouting through laughter and squeals, "You're *it*!"

Math class is ending. A cheer erupts as the teacher tells her students to put their books away. She splits the class into teams. In twos, they approach the chalkboard and faceoff for the love of numbers and grade school honor.

It's Saturday night as a roomful of suburban mothers in their forties gather to play a game of Mahjong. As the tiles click and scores get recorded, they laugh, complain and bond.

It is no wonder that the simple idea of a game can induce some of life's strongest and most satisfying memories. Once relegated to the fringes of our lives—the downtime, the fun between the drudgery of work, the opposite of real life—was the "game."

But the tides are turning. Games have begun a slow bleed into the everyday. They affect everything from how we vacation to how train for marathons, learn English and manage our finances. What we once called play at the periphery of our lives is quickly becoming the way we interact. Games are the future of work. Fun is the new "responsible." And the movement that is leading the way is *gamification*.

Gamification

Bandied about as the marketing buzzword of our time, gamification can mean—and has meant—different things to different people. Some view it as making games explicitly to advertise products or services. Others think of creating 3D virtual worlds that drive behavior change or a method for training users in complex systems.

They are all correct. Gamification brings together all the disparate threads that have been advanced in games for non-gaming contexts. In this way, we unite serious games, *advergaming* and games-for-change into a cohesive worldview that's informed by the latest behavioral research and social game success. For our purposes we will define it as follows:

Gamification
: The use of game thinking and game mechanics to engage users and solve problems.

This framework for understanding gamification is both powerful and flexible: it can readily be applied to any problem in need of a solution.

Take broccoli consumption—there are a lot of children in the world that see broccoli as a real problem. In fact, 70% of us have a gene that makes it taste bitter. This genetic adaptation (found on gene Htas2r38) is likely linked to the fact that cruciferous vegetables (which include broccoli and cabbage among others) historically blocked the uptake of important chemicals to the thyroid. Thus, our perception of bitterness in these vegetables actually once protected us.

It took about ten thousand years to domesticate these vegetables so they became safe to eat. However, statistics show that it takes the average child twelve years to go from hating broccoli to loving it. And research shows that if you possess the bitterness gene, you still perceive the bitterness—even into adulthood. So what has changed? Certainly not the broccoli-eating taste buds. Yet something is different and that difference lies in perception—Bitter is no longer bad.

Similarly, there are numerous online sites that help marketers connect with website developers. From major sites like oDesk to specialized ones like rent a coder, competition for customers and the best practitioners can be fierce. Once the novelty of marketplaces wears off, how do the respective parties decide to choose one over the other? How do the markets ensure loyalty and engagement smog their fickle and price-conscious users?

One such marketplace—DevHub—thinks it's found the answer: gamification. By deploying some of the basic tenets of our discipline, and with the judicious use of game mechanics like points, badges, levels, challenges and rewards, DevHub has quickly differentiated itself as a market leader—raising various engagement metrics, such as time on site, by as much as 20% over pre-gamified levels. With a clear emphasis on making things more fun and rewarding, DevHub has broken the dour cycle of quoting, bidding, coding and follow-up necessary to run a successful web project.

Make no mistake—the core work is unchanged—nothing has fundamentally shifted in the mechanics of designing a website. Only the perceptions of DevHub's users have changed. Understanding our potential to experience the same things in two ways is the first step to understanding the power of gamification.

Engagement

The term “engagement” in a business sense indicates the connection between a consumer and a product or service. Unsurprisingly, the term is also used to name the period in a romantic couple’s relationship during which they are planning to spend the rest of

their lives together. Engagement is the period of time at which we have a great deal of connection with a person, place, thing or idea.

There is no single metric on the web or in mobile technology, for example that breaks down or sufficiently measures engagement. Page views and unique viewers don't quite answer the question of: Who is engaging with our products, services, ideas, websites and businesses as a whole?

Rather, we'd be better off thinking of engagement as being made up of a series of (potentially interrelated) metrics that combine to form a whole:

- Recency
- Frequency
- Duration
- Virality
- Ratings

Together, they can collectively be amalgamated as an "E" (or engagement) score.

The importance of "E" is obvious given that the current prevailing theory—and what is being proved as we move toward a more peer-to-peer, viral and social marketing environment—is that traditional brand marketing isn't really working anymore.

Rather than the antiquated idea of pushing consumers to "buy, buy", more then trying to foster engagement, engaging users first to generate revenue is the marketing model of the future. Simply put, engagement does not follow revenue. But behind engagement, revenue follows.

This is proven clearly in the model of hugely successful social games companies—like Zynga—and one of their key innovations in the field of marketing. They view customers in terms of a funnel, with the general population at the top. Those users are generally not paying to interact with a product, service, or brand. As they progress down the funnel, users are selected based on engagement, with their corresponding spend and commitment to the experience increasing exponentially. In this model, the most loyal customers pay the most, while the average, or novice user is being slowly drawn into the ecosystem. It is a reversal of the classic customer acquisition and loyalty model, and a very powerful view.

Loyalty

The word most frequently used to describe engagement, particularly in a marketing context, is loyalty. In fact, to a great extent they are synonymous. However when you hear the word "loyalty" it conjures up several meanings. The first is easily the loyalty of a dog toward its master—an unfailing obedience that allows a dog owner to do no wrong in the eyes of a pet. Where loyalty and engagement overlap, blind acquiescence is not the kind of loyalty we're interested in developing throughout this book.

What we will look at is a form of loyalty that gets users to make an incremental choice in your favor when all things are mostly equal. When products, price or place are grossly unequal, gamification—and the loyalty it engenders—are much less meaningful. But as in most markets, like web design, eventually commoditization occurs, and it's the bond between customer and brand that drives decisions.

As with broccoli and children, if given enough time and incentive, we can overcome our natural programming. Not to put too fine a point on it, but why wait?

What Gamification Isn't

As we begin our journey into what gamification can do, we also need to be clear about what it cannot do, at least in the scope of this book: gamification is not merely about slapping some badges on your website; a more thoughtful approach is what's advocated here. Also, if you expect gamification to fix your business' core problems—bad products, or poor product-market fit—it will not.

Moreover, this book will not help you build a world where your consumer's avatar is chasing gremlins with an MK-47 in order to save the spaghetti sauce your company is trying to sell. It will also not teach you how to build a Facebook game where users match colored jewels to get discounts on insurance. While these may be viable options for some businesses (in 2003), we posit they are not really the best techniques for building long-term engagement or loyalty with game techniques. Simply put, building actual games-with-a-capital-G is not this book's purpose.

Instead, we will share an understanding of the design process used by some of the world's biggest brands and hottest startups to gamify their customer interactions. We'll start by looking at what drives users to play and the core psychology that makes games so compelling. We'll separate the wheat from the chaff in game design rubric and share what's relevant from the discipline to you, the builder. And finally, we'll show you—in concrete terms—how to architect and implement various core elements of gamification on the web, including some insights from the world's leading platform providers.

Our objective is to give you the tools, techniques and process thinking you'll need to design gamification into your unique experience. It's not unlike learning how to bake—and a cake metaphor is apt considering the dialogue about gamification today. While we can spread gamified "icing" on your product or service with relative ease, unless the underlying cake is also delicious, most users won't want to take a second bite. Exactly the way a great baker creates through the interplay of structure and sweetness, so too must a well-designed gamified site marry substance with reward.

To do so, we will explore how baking gamification into your business from the ground up when paired with a keen understanding of your customer can produce the ideal product. Through the basics of gamification, player motivation, game mechanics and their implementation, you will be handed the recipe that will take your business from

everyday to gamified—from summer heat to a game of tag, from Math class to game time and Tuesday night to family fun.

So put on your apron, and hold on to your toque. Gamification is about to change everything.

CHAPTER 1
Foundations

To begin working on gamification for your product or service, it is useful to begin by throwing out some ideas that might be rattling around in your head. These misconceptions about gamification can interfere with your ability to design with excellence.

As we mentioned in the previous chapter, game mechanics cannot solve fundamental business problems. It will not rebuild poor infrastructure, nor will it heal disastrous customer service. And, unless your actual business purpose is making games, it is unlikely that the result of gamification will give your product the viral power of Zynga's Facebook games like Farmville or Cityville.

As you arrive at the concept of gamification, you might bring with you even more preconceived ideas. For example, perhaps you believe that location-based services like Foursquare serve no real purpose beyond their game elements. Simply put, Foursquare allows players to "check-in" to locations using mobile devices and in doing so the player can earn badges. If they check-in to a location more than any other player, they are deemed "mayor" of the establishment—and recognized as such by fellow players, the business establishment and the game itself. But as we delve even more deeply into this sweeping phenomenon it will be clear that there is more on the line than badges and mayorships—but rather a desire to be connected that drives the user's location-based journey

To some extent, it is the sheer simplicity of Foursquare and similar games that have made them successful. Gamification can fix large scale, complex problems but that doesn't mean its application need be large scale and complex. Gamification that is simple, rewarding and fun can be equally or more effective. And in focusing on game mechanics that are simple, rewarding and fun, you will be amazed by how much can be accomplished.

Finally, for the purposes of this book, we are going to refrain from using the terms "customer" or "user" and instead use the word "player" from this point forward. By thinking of our clients as players we shift our frame of mind toward their engagement with our products and services. Rather then looking at the immediacy of a single fi-

nancial transaction, we are considering a long term and impactful union wrapped in a ribbon of fun.

The Fun Quotient

Let's start here: Everything has the potential to be fun.

Perhaps you are thinking, "No way. How about going to the dentist? Going to the dentist isn't fun!"

Or maybe you first thought is, "Waiting in line is boring. Waiting in a line is the opposite of fun."

We're sure you can think of an endless array of items in this life that are just not fun: Surgery, for example, changing a baby's dirty diaper, clipping someone else's toe nails —to name a few. However, some of the most popular games of the past five years have used incredibly banal ideas as their thematic hooks. In fact, four of the most popular games in the past decade include such fun activities as planting crops (Farmville), waiting tables (Diner Dash) diapering a baby (Diaper Dash) and doing other people's hair and nails (Sally's Salon).

Another highly rated online game has its players perform one of the most stressful jobs in our society—a job that boasts one of the highest suicide rates in the entire world: Air Traffic Control. In the blockbuster game Flight Control players are expected to guide airplanes safely to a runway without killing any of their hundreds of passengers.

So the question is, why did those brazen game designers make a pitch for those games to a room full of executives? And how didn't every single one of them get laughed out of the building? The answer is simple: Fun and the themes of fun are not always connected.

At any casino in the world, a player is faced with a myriad of slot machine faces. From Oprah to Vanna to Harley-Davidson, slot machine imaging looks as outwardly different as a talk show does to a game show or either to a Harley Hydraglide. But they are not different. In fact, other than the logo, those machines are almost identical mechanistically. Push the button, pull the arm and let the cherries align—to win. With all due respect to Wheel of Fortune, it is not the logo that keeps players playing at those machines, but the underlying mechanic.

This does not mean that the brand is an unimportant feature. In fact, it is the way we dress the mechanic that gets most people to pull that lever in the first place. While some people might think that nearly killing hundreds of imaginary passengers in an air-traffic control related incident is as exciting as it gets, others will be drawn in with rippling, muscled heroes and the heroines of a Harlequin romance novel. It might be the underlying game mechanics that keeps the player playing, but the thing that brought each of them into the experience was different—and more than likely, it suited a specific purpose or peaked a particular interest.

Fun is Job #1

In the past 20 years, there have been no major blockbusters in educational software/games, the field otherwise known as edutainment. That's right. Software focused on children, the demographic with the biggest claims on fun, are not getting it where they arguably need it most—in learning. Does this mean that it's impossible to educate using fun? Is school forever consigned to be boring?

The famous geography game, "Where in the World is Carmen San Diego?" was the last big hit in educational games. It was inarguably a tremendously fun way to learn about country and province capitals, as well as the major exports and waterways of places thousands of miles from the classroom. Since then, thousands of educational software companies have been started and failed.

So where in the world is the next big hit? Games aligning entertainment and education like Sim City and Civilization have taught hundreds of millions of people history lessons and the basics of urban planning. These are not pedagogical games. They weren't designed to be educational. But they use history and real city schema as a backdrop to explain ideas—thus education is a byproduct of fun.

Which is precisely the opposite of what has happened to educational software. In fact, what happened there was very simple—teachers and parents got involved and systematically extracted the fun from the game. Kids could smell that shift from fun to work from a mile away.

So, can children learn from games? It would seem that they can. Will they learn from a game if it is not fun? Judging by the state of the educational software industry, they will not. In other words, if you start with the education and put fun second, learning doesn't seem to work the same way—or as well.

The Evolution of Loyalty

Loyalty, as we've already mentioned will be defined for our purposes as encouraging an incremental choice in your favor when all things are equal. Loyalty and consumerism share a long and varied history together. While there are likely ancient examples of loyalty programs in one form or another, we will begin with loyalty programs in America.

With the growth of urban centers in the 19th century, markets and local mercantile establishments were beginning to thrive. People arriving in town to buy their weekly supplies were often plied by one seller or another to buy ten pounds of sugar and get their next pound free. Thus a 10:1 ratio came to form the basic structure of loyalty programs. In fact, it is such a canonical example that 95% of all loyalty programs today remain buy-ten-get-one-free.

The fundamental problem with this model is that it gives the people most likely to pay you anyway, things for free. This is a fundamental flaw that social game designers do

not abide. While there is nothing wrong with offering a reward—or a "thank you"—to your most loyal players, their purchasing habits might not be negatively affected without the freebie—but new, or novice customers, certainly are.

The 10:1 model remained the standard until the 1930s when S&H Green Stamps was launched. S&H's program had participating merchants reward players with stamps in exchange for specific purchases. Those stamps were then put in a book and those filled books were saved up. The books of stamps could be traded for free stuff from a catalogue or at an S&H Green Stamp store at various rates, depending on the desired product.

What Green Stamps understood was that with the introduction of a virtual currency, people lose track of value. They can no longer identify the how much those individual stamps are worth. Whereas—buy 10 pounds of sugar, get one free exactly values that free pound of sugar, with the advent of Green Stamps valuation was vague: A shirt is worth 16 Green Stamps but pants are worth 20. A transistor radio is valued at 60, but a trip to Hawaii is 6,422...so how much is that stamp worth, again? So while consumers had little concept of value exchange, S&H knew exactly how much those stamps are worth. Thus, the first virtual currency was created.

Loyalty programs continued in a similar fashion until 1981. First American Airlines introduced AAdvantage, followed in short order by United's Mileage Plus and TWA's Aviator. It was with the development of the frequent flyer program that businesses discovered that loyalty is less about free stuff than it is about status. And if you've ever tried to redeem miles for a trip to Europe in the summertime, you understand immediately that the free flights are not the core of the value proposition of the system. In fact, people are quick to make the connection between standing in a faster line to get their baggage screened and joining one of these programs.

They also understand that players get better places to sit, faster phone and Internet customer help and better overall service. None of these things cost the business a cent, but each one is a powerful driver of brand loyalty for their players.

Frequent flyer programs remained the best loyalty program model until recently, when virtual rewards systems began popping up online and on mobile. A game like Farmville doesn't even pretend to offer real-world prizes. There are no faster lines, no five books of stamps for a model airplane, no free bags of sugar. In fact, the notion of redemption of any kind has been dropped altogether. No redemption. Not a pin, not a cup of coffee. Nothing. The cost of producing customer loyalty has dropped precipitously. It used to be 10 cents on the dollar—10%—buy ten get one free. Now, for applications like Farmville or Foursquare, costs are nearly zero.

At the same time, the value of status choices is rising. So in the old days (pre-2008) if a person had a preference for Cuisinart over KitchenAid, for example, how was that expressed? How did he get his friends to understand his loyalty choice? First of all, the friend had to be standing in the kitchen near the product itself. Then, a conversation would have to introduce it. This process was called word of mouth marketing.

In short, the lesson of the word of mouth marketing industry was: build a great product and consumers will talk about it. There was no process, and word of mouth marketing was a hit driven business.

Alternatively, a company like Zynga has a very good idea for how their word of mouth marketing works. It's part of the phenomenon of social networking. On Twitter and Facebook, players of Farmville constantly express their loyalty. If your product is not in this "loyalty stream" on social networking sites then it is not even a part of the discussion. Loyalty is no longer private. It is no longer standing in a kitchen beside your favorite mixer. It is public, and millions are viewing it.

Status At the Wheel

In the fourth season of the show, Deal or No Deal, a young woman finally won the coveted million-dollar prize by randomly selecting the suitcase in which it was contained. If asked, you probably couldn't name her off the top of your head even though it took four seasons on a popular game show to get that win.

A second reality show contest winner has a decidedly more familiar name—Christian Siriano, fashion designer and winner of the fourth cycle of the Heidi Klum hosted series, Project Runway. If you don't know his name, perhaps you know his catch phrase—When displeased with something, Siriano refers to it as "a hot mess," a phrase that has since entered the American vernacular.

Where the million-dollar winner of Deal or No Deal came and went as quickly as that suitcase was opened by a buxom-blond, Christian Siriano's name has remained. What was his prize? $100,000. With his winnings he can't afford to start a fashion line in New York City, Paris or London. Let's be honest, he can't even afford more than a few items of high-end fashion purchased off the rack for that amount of money.

Similarly, the winners of Top Chef, Hell's Kitchen and Chopped all compete for negligible cash prizes—A Chopped champion bringing in a mere $10,000 minus taxes. So, what are these people playing for? It is unlikely they are taking themselves away from their families, putting themselves through difficult and rigorous competition over the course of weeks in order to win a little money and some steak knives. So what is the drive? What did Christian Siriano get when the camera's turned off?

After the show, Siriano went on to design a fifteen-piece collection for Puma, develop a make-up line for Victoria's Secret, and write a book. While his stint on Project Runway did not afford him riches, it did win him fame, recognition and status—in spades.

If you don't have a ton of cash to give away as an incentive (who does?), status is an excellent alternative. It is a great driver of loyalty, not to mention a player's fiscal behavior. And over time, you can bet it is a whole lot cheaper. A gamified program with a status benefit needs far fewer monetary, physical or even real-world-redeemable re-

wards. Status is, as American Airlines understood in 1981, and most of us clearly grok today, an extremely powerful reward. But is it everything?

SAPS

SAPS is an acronym that stands for status, access, power and stuff. Simply put, it is a system of rewards. Conveniently it lists each potential prize in order of the most desired to the least desired, the most sticky to the least sticky, and the cheapest to the most expensive.

Status

Status is the relative position of an individual in relation to others, especially in a social group. Status benefits or rewards give players the ability to move ahead of their peers or strangers in a defined ranking system. Importantly, this ranking system need not be based on the real world at all—it works perfectly in a purely constructed environment. Some examples of status items can include badges and leaderboards.

> Though we talk about them in greater detail in future chapters, here is a brief introduction to two core mechanics that affect and measure status.
>
> *Badges*
> : Badges are a known status item. They can be given out virtually or physically. However they must be visible to other players in the game or else their meaning and valuation is limited.
>
> *Levels and Leaderboards*
> : Levels and Leaderboards are another way to indicate that a player has more or less status in a given game and can be a powerful tool in your quest for loyalty.

Access

Gilt Noir is a social website geared toward flash sales of high-end fashion. The top 1% of Gilt players receive a scented candle and card in the mail. Other than this $20 package, these top players get a fifteen-minute head start on everyone else for all sales. For Gilt, the prize doesn't cost a thing. But to the player, that extra time to pull in the best bargains is exceptionally meaningful, as supplies in each sale are limited.

Consider how seemingly revolutionary this is in the context of most loyalty Programs. Instead of offering top customers discounts or giveaways, gilt noir members are given early access. This is a process-driven version of the informal Programs long at work at high-end fashion houses like Bottega Veneta or Gucci, where top buyers and the famous get first dibs on cool new products.

Other ways to provide access as a reward to your players can include a sit down lunch with a CEO or editor. Maybe your players would find it interesting to sit in via Skype

on a company meeting. Or they might receive priority or VIP seating and the earliest possible appointments.

Power

Prizing power to your players offers a modicum of control over other players in the game. A good player might be asked to serve as a moderator on a forum, for example.

Not only will players do work for you for free, to them power benefits appear huge. Most forums and World of Warcraft successfully offer positions of power for which their players vie on a daily basis.

Stuff

While this list indicates that "stuff" is the least important reward or prize, we are not anti-freebies. If you have great items to give away, and for players deeply expecting to receive free items, stuff can be a strong incentive.

Once the item has been given away, however, the incentive to play is finished. In other words, stuff is only good until it is redeemed and that is the exact length of time your players will engage themselves around the game.

Of course, someone might argue that they'd rather get, for example a free ice cream than be badged "Ice Cream King." And off the cuff, almost anyone would agree with that assessment. However, it is important to remember that "off the cuff," no one is yet in the game. Once she is, the value of becoming a "Ice Cream King" might mean that she has reached a new level in game play that allows the her to enter a contest to design a new ice cream flavor. Or maybe it will allow her to skip to the front of the line every time she comes in to buy an ice cream cone.

It's always the depth of meaning of the game that matters, not the monetary value of the prize. Remember, no matter what that tangible prize is, you need to disclose its value (or it's inherently known to your most loyal customers)—so they tend to value the interaction accurately. Say that a regular coffee drinker just earned his 11th latte for free. After buying 10 of them for $2.50 each, he probably knows what the 11th is worth.

But what value do players attribute to status, access and stuff? They cannot accurately price those benefits and—in general—tend to over value them. For example, when assessing the importance of not having to wait in line, most people over value their time saved. Similarly, they don't know how to quantify the six minutes they got to meet and chat with Lady Gaga backstage. But the game designer does, and it's almost always cheaper—and sticker—than giving them free stuff.

The House Always Wins

And that truism underlies the last basic lesson of games in the real world: No matter what the player thinks, the house will always win a well-designed game. Like any casino manager will tell you, while the illusion of winning is vital to motivating use and play, actually winning is much harder than it seems.

Broadly speaking, this has implications not only for players, but also for those of us charged with building and designing great user experiences. As markets gamify and consumer demand for fun, engaging and creative experience increases, you have a fundamental choice: either be the house, or get played.

Trust us, you want to be the former.

CHAPTER 2

Player Motivation

At the root of Gamification is the player. In any system, how the players are motivated ultimately drives outcomes. Therefore, understanding player motivation is paramount to the building of a successful gamified system.

We already know that games are generally good motivators. By focusing on three central components—pleasure, rewards and time—games have become one of the most powerful forces in all of humanity. Uniquely, games are able to get people to take actions that are not always in their best interest, without the use of force, in a predictable way.

Powerful Human Motivators

From Greek Mythology to the daytime soaps, it is clear that sex—or the drive to have it—will make a person do almost anything. Paris' abduction of the lovely Helen of Troy led King Menelaus to begin the Trojan War. So, like games, sex does have the unusual ability to make people do things that are not in their best interest. However, like the Trojan War—and unlike games—it is not a predictable motivator.

Similarly, violence can yield unparalleled motivational results. Putting a gun to a person's head will likely get him to accomplish any task you want him to accomplish. However, chances are, he won't enjoy a second of it—and certainly won't come back for more. The force fallacy, that punishment can accomplish great results, is a powerful, flawed belief.

Games, however, hit the sweet spot. They marry the desire-drive of sex with the predictability of duress—except without force and, when successful, driven entirely by enjoyment. This pattern is also why games have a dark side: People addicted to slot machines can look like they haven't seen the sun in months, while World of Warcraft players are sometimes accused of neglecting their real-life duties for the sake of the game. But there is also a bright side to games—a side in which they are changing people's health, the way they learn and the way they live for the better.

Flow

At the heart of the success of games is something called *flow*. Our understanding of flow is derived from the research of Mihaly Csikszentmihalyi, a Hungarian-born American psychology professor who is noted for his work in the study of happiness and creativity. Achieving flow—or being "in the zone"—indicates the state in which a player is between anxiety and boredom and meeting their own motivational level in that experience.

When a jazz player is playing music, or a runner is training, they exist in a state of suspended animation. They are calm and focused. A writer, mid-stream in narrative forgets, for the moment, the outside world. It is safe to suggest that almost everyone has had that experience where they lose track of time and space while playing a game, cooking, working out, cleaning their homes, or talking on the telephone.

Meanwhile, game designers are obsessed with creating this state for their players. They are always looking for ways for the player to be at one with the game. It is a constant quest to bring someone into their system only to leave her, seamlessly, in the highly prized state of flow—but how?

Through the careful interplay of system and player, and a relentless belief in testing those interactions to find that point between anxiety and boredom. And with a range of other psychological phenomena to choose from, such as reinforcement.

Reinforcement

If a mammal such as a lab rat is provided a pellet of food at a fixed interval every hour —during the 59 minutes between each pellet, the animal will invariably go off and do something else in its cage. Only at the 60th minute, will it come back to get the dispensed pellet.

This structure is similar in form to many Industrial Era jobs. A worker gets a paycheck every two weeks. What happens in between is completely aligned with that end result. In other words, the worker will only enact exactly what is required of her to ensure that they will get their bi-weekly paycheck during the days in between. No more, no less. This is called fixed-interval reinforcement. Not surprisingly, fixed-interval reinforcement schedules tend to yield low levels of engagement.

On the other end of the spectrum there is variable-ratio, variable schedule reinforcement. In this model, the lab rat doesn't know how big the reward will be or when it will come, but at some point it knows it will come. Therefore that rat will press the dispensing pedal in its cage endlessly until it gets its reward. It is exactly the model used in slot machine gaming as well as for almost every other archetypal gambling model. Another name for this behavior modifier is operant conditioning and it is undeniably addictive to mammals.

While operant conditioning experiences can be dangerous and turn offs for many, their judicious use within a broader game-like experience is a powerful force for driving player behavior.

But unlike rats in a cage, most humans are not required to play—in fact, that kind of design can be a huge deterrent to engagement. So why do we play in the first place?

Why People Play

A good working theory for why people are motivated to play games maintains that there are four underlying reasons, which can be viewed together or separately as individual motivators:

1. For mastery
2. To de-stress
3. To have fun
4. To socialize

In a 2004 paper entitled "Why We Play Games," Nicole Lazzaro, an expert on player experience and emotions in games, outlined four different kinds of fun.

Hard Fun: where a player is trying to win some form of competition
Easy Fun: where a player is focused on exploring the system
Altered State: in which the game changes the way the player feels
Social Fun: during which the player engages with other players

In 1964 the famous social science best seller, "Games People Play" by Eric Berne, M.D. exposed games organically cultivated through social interaction. The book, focusing heavily on the social engagement of "housewives," (an arguably outdated construct to be sure, but en vogue for its time) managed to recognize some interesting insights about social game play. In one of the book's most compelling examples, Berne talks about a game engagement between the women whereby they go around a circle and talk about the ways their husbands upset them.

"He leaves his socks all over the place," one bemoans. "He hates my cooking," says another. "He forgets my birthday," laments a third.

But what would happen if someone in that circle chose not to play the game? According to Berne's research, if a fourth housewife says, "Actually, my husband is a good guy," there is a consequence to her action that is decisive and chastising. At the next party, the fourth housewife won't get invited back—plain and simple. The prize in this game is to win a follow up invitation.

The interesting point that Berne leaves us with is an emerging understanding of the social games that exist organically in the strata of our society. Even language indicates that we've been aware of these "hidden" games for some time: a person is called a "player" when they can get a lot of dates in the "dating game." A government official

"runs" for office in a "contest." Someone who is socially unsuccessful is labeled a "loser."

EXERCISE

Your Player's Story

Visit GamificationU.com to download the exercises and unlock bonus materials!

Write the story of the canonical player of your game. As you design your system you will focus on this player and what might appeal to him. What is his story—from his demographic to his psychographics? Write in one paragraph who he is.

Example Players

Susan is a thirty-two year old fifth grade teacher who lives in an urban community. She is mostly able to live within her means, with the help of a credit card and occasionally, her parents. She enjoys going out to clubs and concerts. She cooks for her friends and hosts dinner parties whenever she can. She is very fashion conscious and easily spends a quarter of her paycheck on clothing and beauty products. She is obsessed with Facebook.

Chris is a twenty-two year old recent college graduate who lives with three roommates in a suburban city. He works at a local radio station as a paid intern and part time as a line cook at a local bar and grill. He is in a band and is definitely interested in girls, although he'd trade a date in a second for any of his "boys." His favorite possession is easily his iPhone, although he would never admit it.

Ron is a fifty-three year old salesman who listens to Rush Limbaugh or NPR—depending on who is telling the more entertaining story—during his many hours on the road. He also enjoys listening to Jazz. He and his wife are planning to take a tour of wine country this fall. He is hardworking and motivated. He has two daughters who are doing well academically, one of whom is going to graduate high school at the end of the year. He worries about paying for college although he's been saving for it since his children were small.

Player Types

The more you know about who is playing your game—both current and prospective, the easier it is to design an experience that will drive their behavior in the desired way.

One rubric that can help you understand your players is to leverage the work accomplished by Richard Bartle in understanding player types. In his seminal work, developed by studying players of MMOG's, Bartle identified 4 types of players. Since then the number has expanded from four to eight to sixteen, however, these four remain arguably the stickiest and therefore the most interesting for our purposes:

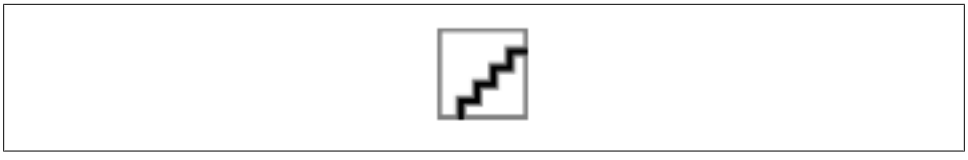


Figure 2-1. Bartle's player types

Explorers
: An explorer, in brief, likes to go out into the world in order to bring things back to their community and proclaim, "I discovered this thing!" One example of a game suited to the explorer player type was Super Mario Brothers on the Nintendo Entertainment System. A player had to play 100 games or more to find every hidden level behind every pipe and block and bring it back to her peers for kudos.

Achievers
: People who like to achieve are an integral part of any competitive game. They drive forward a great deal of projects, services and brands. The problem with designing exclusively for this player type is that it's difficult to develop a system where everyone can win and achieve. And for achievers, losing at the game will likely lead them to lose interest in playing it.

  Moreover, a common bias your authors have observed in working with clients to gamify their experiences is that the majority of system, site and product designers are high-achieving people. So you naturally infer that the majority of players are similarly inclined. This turns out not to be true at all. The majority are socializers.

Socializers
: This player type is made up of people who play games for the benefit of a social interaction. Games focused on socializers make up some of the most enduring games throughout history—Dominoes, Bridge, Mahjong, Poker—the thread tying them all together is that each is an extremely social experience. But to be clear, it isn't that socializers don't care about the game or winning—they do. However, the game is a backdrop for meaningful long-term social interactions. It's the context and catalyst, not the end in itself.

Killers
: Killers, also known as *griefers*, make up the smallest population of all of the player types. However, they are an important one to understand. Similar to achievers in their desire to win—unlike achievers, however, winning isn't enough. They have to win and someone else has to lose. Moreover, others need to see it happen or it wasn't really a win.

Bartle did not intend these four player types to serve as a personality inventory—they weren't developed with that in mind. But with decades of game thinking under our belts, it is easy to see how they can be useful when considering the players of gamified systems. And, by arranging the player types on an axis, we can see how they range from acting to interacting and from people to environments.

Note: People are not exclusively one or another of the four player types. In fact, most people have some percentage of each. In all probability a person's dominant player type changes throughout his or her life—and even varies from game to game. But it is a compelling way, as a game designer, to see how people are motivated to play and interact in a gamified system.

> Go to *Http://Bit.ly/Bartletype* and find out what player type you are.

If you take Bartle's test to figure out your player type, you will notice that, as we mentioned, they are mutually inclusive. In other words, a player can be 100% explorer, 100% achiever, 100% socializer and 100% killer, all at the same time.

Most people are not. For the average person the breakdown might look something like this:

- 80% socializer
- 50% explorer
- 40% achiever
- 20% killer

If the scores were mutually exclusive, however, and a player could only be one or the other, we learn that the vast majority of people are socializers: as much as 75% of us. In the context of the runaway success of games like Farmville or poker, that statistic should not come as a particular surprise. Explorers and achievers make up about 10% each of the population and killers 5%.

By bringing together many of the understandings we've developed in this chapter, the underpinnings of the social game movement suddenly become more apparent.

Social Games

The video games that have informed so much of our game thinking over the last few years are actually the exception and not the rule. It wasn't until runaway hits like Farmville and World of Warcraft (WoW) that game thinking began to bring out the social element. This is despite that fact that most of our games historically have been social. In fact, nearly all of our games throughout history—with the obvious exception of Solitaire—have required other people to play.

Video games came about as the first real single player games. So for most video game developers, their view in terms of the importance of socializing in games is distorted. The last 35 years have been about designing single player games. So the success of a simple social game like Farmville seems like an amazing breakthrough even though it's actually the status quo.

The truth is, game developers and technical designers are unlikely to find themselves in the norm when it comes to the players for whom they are designing. Like you, they

tend to be an achievement-oriented breed, more so than the average person. It's a big bias to overcome. More than likely, when they are thinking about game design they are amassing points, seeking status, even killing to win. But they are not the average person.

The average person is looking to socialize, not win. Winning is not what drives society. If designers begin by thinking the game is about achievement, they will at some point realize they are wrong. The average WoW player is as dependent on the guild (team) with which he fights his battles as he is on the battles themselves. Excusing himself from a family dinner to avoid letting down his team for a 7 p.m. raid is far greater a motivator than the raid. The raid by itself could wait until after dinner. Most of society plays for the camaraderie and community of the game more than for the win.

EXERCISE

Rank Your Top 5 User Actions

Rank the 5 user actions you most want your players to take (You can add your own!)

- Advocate
- Argue
- Comment
- Compare
- Compete
- Curare
- Explore
- Express
- Flirt
- Give
- Greet
- Harass
- Help
- Join
- Like
- Poke
- Rate
- Read
- Recommend
- Share
- Show off
- Taunt
- View

- Vote
- Example Ranking of Your User's Actions
- Using an auction website as our example:
- Compare
- Compete
- Explore
- Rate
- Show off

Once you have ranked the five most important user actions to your service (an auctioning of goods) place them on Bartle's chart. Where do the actions overlay on that chart? In the above example, the list seems to focus on achievers/killers (compete, show off), socializers (compare, rate) with a little bit in the explorers' category (explore).

For an auction site it makes sense that this is where most of the players fall. Collectors want to win. Sellers want you to "win" for as much money as possible. Especially in the last minute, an auction is very achievement/killer oriented.

A good rule of thumb: Take note if you don't have any actions in the socializer quadrant. You are probably missing something about the experience. Socializers are the most universal. You may not always have a lot of players who fall in the killer quadrant, but you will always have socializers.

If you are running a site dedicated to music, perhaps your list of user actions and where they fall on Bartle's chart would be markedly different. You might find that more of your action points serve socializers and explorers rather than achievers and killers. Music is an inherently social activity. But an explorer is drawn to the component of music that allows her to declare to her peers, "I discovered this musician."

Hint: If while doing this exercise you don't have a particular product or service you are developing, we suggest you imagine that you are creating for traders on floor of the Chicago Board of Trade. Here you will find a very achievement oriented population. But you will likely discover that there are still necessary social action points, even when the achievement or killer instinct is high.

Intrinsic vs. Extrinsic motivation

Another aspect of understanding player motivations is the question of "where do motivations come from?" broadly speaking, psychology has divided our motivations into two groups: intrinsic and extrinsic. Intrinsic motivations are those that derive from our core self and are not necessarily based on the world around us. Conversely, extrinsic motivations are driven mostly by the world around us, such as the desire to make money or win a spelling bee.

When it comes to intrinsic vs. extrinsic motivation in a gamified or motivational design context, there are three schools of thought:

In the book Drive by Dan Pink, Pink attests that cash is weak reward for getting players to complete complex tasks. The research he rounds up shows how an extrinsic motivator like cash doesn't work when people are given lateral-thinking tasks. In other words, when cash is introduced as a motivator, people's performance on creative or complex tasks drops. Thus, he contends that cash rewards are bad for incentivizing creative thought.

While we agree that cash is not always motivational and in some cases is actually demotivational, other extrinsic rewards can be astoundingly effective at motivating players. For example, long-term social status rewards can be particularly effective at nurturing creativity and play. Although bad extrinsic motivators might be ineffectual, good ones are not.

Dr. John Houston, a preeminent researcher in competitiveness, found that exceptionally competitive people are self-destructively competitive. His research showed that people—principally achiever/killer types—with a high level of competitiveness compete even when there is nothing to be gained. Moreover, they tend to complete even when there's a clear disincentive to do so. When told that they must collaborate with a partner, hyper-competitive people will continue to try and figure out how to win, even against a collaborator, even when there is nothing to win.

Overjustification/replacement bias argues that replacing an intrinsic motivation with an extrinsic reward is a fairly easy thing to do. Research suggests that when a child who plays the piano simply because she enjoys it is introduced to competitive piano playing, many changes in her behavior can occur. For example, if she begins to win badges and trophies and subsequently not win she will stop playing piano. That is, we crush intrinsic motivation with extrinsic rewards and she never comes back.

So once you start giving someone a reward, you have to keep him in that reward loop forever. Because when you take it away, he will no longer possess the level of intrinsic motivation with which they began.

EXERCISE

How Competitive are you?

Take the Houston and Smither's Competitiveness Test to measure how competitive you are.

Insert Test

Old Beliefs

One of things that continues to come up in all of this research on competitiveness, overjustification and incentives and rewards is a series of old beliefs that we've held onto that are thought to lead up to the transformative power of gamification. However, we'd like to look at them more closely.

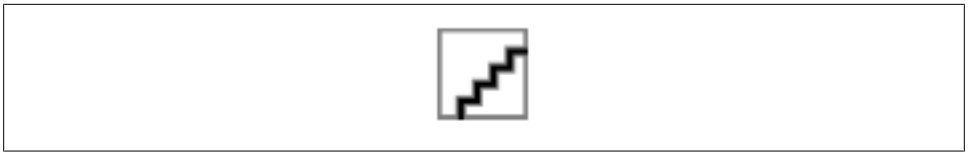


Figure 2-2. Priority boarding lane

Old Belief #1

Is intrinsic better than extrinsic?

Does it even matter? Or more importantly, can an intrinsic motivation be counted on? The crux of the problem is that sitting around waiting for people's intrinsic drive to become better, to do more, or to help other people, we find we don't always get the results that we want. But when we make the motivation extrinsic, we shift that locus of responsibility from hope to structure and process. Gamification, meanwhile, is designed to do just that.

While we don't know where intrinsic motivation comes from at all, it isn't unimportant. But it is far less helpful to a game designer designing a game. The new belief is that we should accept our players as they are, perhaps despite their intrinsic motivation—and ply them extrinsically to the best of our abilities.

Old Belief #2

Intrinsic motivators create greatness, while extrinsic motivators are nothing more than pellets dropped for rats in a cage.

Our fundamental observation is that when something is designed well, it feels intrinsic to the player. His perception is that it was intrinsically driven whether or not it was. Just as a good sales person can convince the buyer it was his idea to buy that set of encyclopedias in the first place. Further, a player might not know something was intrinsic to him until he discovered it through an extrinsic motivator.

As designers we must to some extent apply a parental eye to our players. We have to attempt to know some things about our players and their wants that they might not yet know. A good example is the motivation to lose weight—Americans, by and large have a real struggle with obesity. Most people want to be healthier and skinnier. But despite the intrinsic desire, the process is disconnected. A designer, however, would say that what is missing is a well-designed reward and incentive loop for losing weight.

The new belief is that we are helping people, to some extent, reach a higher potential —and to discover things about themselves that they didn't already know.

The best designs are intrinsic: Priority Boarding

It may surprise you to know that Continental (now United) Airlines actually invented the model of the priority boarding lines and carpets. Once upon a time, we boarded over a loud speaker, with the first class leading the way, followed by designated seating

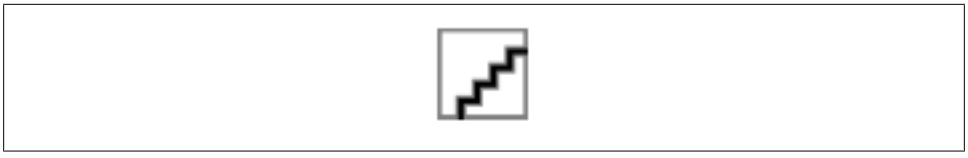


Figure 2-3. Mountain

rows. But Continental realized that for no more than the cost of a carpet and a sign, they could take what was once a private status benefit and make it public.

To be clear, there is no new benefit granted to the people boarding from the red carpet. They always boarded first. They always got the extra overhead bin space, the drinks before take off. In fact, the plane doesn't leave any sooner and the honey-roasted peanuts are not any better than they were before. The only benefit is a demonstrable one. And the only true beneficiary is Continental Airline's.

Now, in addition to flyers being able to show off that they are on the red carpet and everyone else is not, all the other flyers will wonder, "Why is that guy on the red carpet and how do I get there?"

Today, the red-carpet boarding design feels normal. In fact, it feels intrinsic, like it's always been there. But it hasn't. It works because the system in which it resides is very compelling. It proves that when the gamified experience is good enough, benefits can be created out of thin air. For example you can even add a baggage fee—and then remove it for people with high status in the frequent flyer game. Even though three years ago, no one paid a baggage fee, now, with a little added friction it has become, like the red carpet, a reward.

A good extrinsic motivation is a good map to intrinsic motivation. The better a designer knows her players and the better the resulting game design, the less it will feel to the player like being on a wheel, and the more it will feel like it was her idea to begin with. That's the holy grail of gamification: A game well designed enough that the player's actions just feel normal. We believe it can be done in almost any experience.

Progression to Mastery

One well accepted theory is that players in any experience are seeking mastery—A player interacts with a system in order to master an idea. Why do people go to Weight Watchers meetings? It's safe to assume it's probably not for the free coffee, but rather to master their weight and health.

In original research by Hubert Dreyfus for the US Army in the 80's (since revised in the 1990's) a series of stages of mastery emerged when looking at how people engage with systems.

Dreyfus outlines five core levels:

Novice

A novice is someone who's just arrived to the experience. It is his first minute with a new system. Using the example of a 1040 tax form, the first time someone sees it he is a total novice to the system. To him it has little depth of meaning. It's just a form.

Problem Solver

Similar to a novice but with some information already in hand, a problem solver is on her way toward figuring out what is going on. On a 1040 tax form, the problem solver has already learned that the instructions are on the back of the form. She may not understand how the numbers are derived or why she is calculating AMT but she is able to do it—and if she can't, critically, she knows whom to call to find out how.

Expert

An expert has already starting learning how the system works. In other words, an expert user of the 1040 tax form knows he needs all of his W2 forms and 1099s attached to this form. If he doesn't submit all of them, regardless of timing, his tax return may be audited, returned or recalculated by the IRS. That is not an obvious bit of information. Nowhere on the form will it explain to a novice what to do if his bank doesn't send him a 1099 in time. An expert knows to hold on to his tax return until he gets it or he can input the number (and skip attaching the form entirely). At the expert level a player knows something that is not obvious to the casual player.

Master

A master believes that she truly understands the system. She believes that she is in control. She is aware of its nuances and its ins and outs. A master is also likely to identify personally or culturally with the system. For example, with a 1040 tax form, the master might be thinking, "I'm an accountant. I've been doing taxes for 25 years. No one knows this process better than me."

Visionary

A visionary is a special kind of master. He puts himself inside the designer's shoes. A visionary can look at the 1040 form and find that there's a flaw in the calculation on line 17. Most designers have probably experienced the enthusiasm of a visionary —like a 3 a.m. email to announce the great idea someone has for some minute aspect of their system.

EXERCISE

Rank the Top 5 User Actions on the Scale of Progression to Mastery

It is important for a designer to now look at those top 5 user actions and decide where they should go on the scale of progression to mastery—understanding, of course, that not everything can or should come at the novice level.

While no player is obligated, nor should she be expected to progress to visionary. In fact, your system should enable the player to stop at any level. If a player is happy to remain a problem solver, this is a perfectly acceptable level to maintain. No player need get any more knowledgeable about the game than that. In fact, it could be argued that most people, when it comes to the 1040 tax form, remain decidedly at the problem solver level. If they need to go deeper they will call a master in the form of an accountant. In any game system, a player should be able to leave the system any time she wants.

Socializing actions should happen across all levels of the mastery chain. Advocacy can take place at both the novice and expert levels. To some degree, it isn't a bad idea to allow every level exposure to the accomplishments of the others.

For a business built around artistic expression—a photo site for example—the player at the novice level must be able to view site content easily. At the problem solver level, he might rate the photos on the site. But how as a designer can you reveal complexity in a measured way? How do you allow for creation before a player becomes a master? And how does the business maintain the integrity of the art on the site if it allows for people at a novice or a problem solver level to create?

Creation can happen earlier, but perhaps it is focused on curation. After all, curation is a form of creation. For the seller audience on eBay, curation crosses all levels of mastery. By procuring positive ratings, to players at the novice and problem solving levels, the seller gains credibility. On the expert and master level it allows them to play more competitively—measured by how well they stack up against other sellers. Once a seller reaches the thousands/millions of sales, within the game itself, it is just assumed he has become a master or potential visionary.

Designing for the novice, considering the elder

The preceding examples highlight the fact that the game someone is playing at the novice level is different than the game someone else is playing at the expert or master levels. The top levels of a game are sometimes called the elder game. When beginning a new design it is important to focus on the non-elder game, the game being played at the novice and the problem solving levels.

This is because, with few exceptions, most people in the world are not yet your customers, which means they are novices and problem solvers in your system. If you begin with a strong and compelling base design, you will find there is no down side to setting aside the elder game until you need one.

Beating the Boss Level

When United Airline's Mileage Plus frequent flyer program began in the 1980's, it wasn't conceived that players would ever reach the million-mile flown mark. After all, a million miles is the equivalent of over 2,000 hours (or a full time work year) spent in a moving plane, not counting any travel, airport, security, boarding, taxiing or de-

boarding time. So when the first players began reaching that milestone, United pieced together a Million Miler level with lifetime benefits.

At first the level was informal, but then it was formalized in the 1990's. What United discovered quickly was that after those players passed one million miles, they tended to reduce or stop playing altogether. Their best, most loyal players simply leveled out of the game. It was inevitable. Without a continuation of the game, there is no longer incentive to play. In the case of these Mileage Plus players, they had beaten the boss level of the game.

Then, a few years ago, United formalized the 2, 3 and 4 million mile tiers for Mileage Plus. They changed and expanded the elder game as needed. In the beginning, it wasn't important for United to have a three million mile tier. Now it is. Luckily they had in place a system that was open and flexible enough to keep the game moving and expanding as needed.

In another example, Farmville began with seventy levels. More were added when players began finishing out the original seventy. These examples show that you must not lose sight of the elder game because of its potential and hopeful inevitability. However, you need not design for it until you have to.

EXERCISE

Rank Your Own Goals and Objectives For Game Design

You as a game designer should consider how important each of the following is to you as you build your system. Keep them in mind. (Feel free to add your own.)

Managing money

Making and keeping relationships

Career success

Helping others and doing "good"

Being knowledgeable

Being healthy

Other

Motivational Moment: Be the Sherpa

On the journey to mastery it is important to know where your brand, service, product, or concept lives in the eco system. You are not the mountain. You are the Sherpa.

Your player is on her own journey. You must make it your goal to help surface that journey and to put structure around it. You must be the guide up the mountain. That is how you obtain long-term, enduring loyalty and connection from your players. You

don't need to be the mountain and you don't need to create it. You simply need to lead them up.

Be their Sherpa. Give them the status, access, power and stuff that gets them where they need to go. Do it right and they'll forever be yours.

CHAPTER 3

Game Mechanics: Designing for Engagement – Part I

Game design is a relatively new, unaccredited discipline with roots in both psychology and systems thinking. When creating a gamified experience, we leverage many aspects of game design while focusing on the core elements that will produce the greatest impact for our players. For example, we generally ignore narrative structure in gamification, because we are building "non-fiction" experiences. That is, the arc of your gamified system is based on your player's and brand's story—as they already exist.

Luckily, you don't need nor should you want to become a full-fledged game designer. While many reference works (such as the excellent Game Design: a Book of Lenses by Jesse Schell) can help deepen your understanding of how to make games, we've filtered the key elements of the discipline here to focus on the most important aspects. Our view of game design is narrow, but highly optimized for gamification.

MDA Framework

One of the most frequently leveraged frameworks of game design is referred to as MDA —which stands for:

Mechanics
Dynamics
Aesthetics

The MDA framework is a post-mortem analysis of the elements of a game. It helps us use systems thinking to describe the interplay of those game elements and apply them outside of games.

Mechanics make up the functioning components of the game. At their core, they allow a designer ultimate control over the levers of the game—giving you the ability to guide player actions. Dynamics, meanwhile, are the player's interactions with those mechanics. They determine what each player is doing in response to the mechanics of the system

both individually and with each other. Sometimes, game mechanics and game dynamics are used interchangeably, but they are markedly different.

Finally the aesthetics of the system are how the game makes the player feel during interaction. Game aesthetics can be viewed as the composite outcome of the mechanics and dynamics as they interact with and create emotions.

Game Mechanics

The mechanics of a gamified system are made up of a series of tools that, when used correctly, promise to yield a meaningful response (aesthetics) from our players. For our purposes, we'll focus on seven primary elements: Points, levels, leaderboards, badges, challenges/quests, onboarding and engagement loops. We will go over each of these mechanics in detail but we will start with the heart of any gaming system—points.

Points

Points are important whether or not their accumulation is shared between players or even between the designer and the player. When you first think of a point system you might immediately think of a goal in a sporting event, redeemable points in a video game or bonus points awarded to players for successfully completing special tasks within a game.

No matter what your preconception of a point may be, suffice to say, they are an absolute requirement for all gamified systems. As the designer, it is imperative that you value and track every move your player makes even if those scores are only visible to you from your design console and not to the player. In this way, you can see how your players are interacting with your system, design for outcomes and make appropriate changes and adjustments.

Point systems run the gamut from in-your-face obvious to barely visible and serve a wide range of purposes. As such, there are a few types we'd like to point out that you've doubtlessly experienced in your life.

Cash score

This is the number that tracks how much money you have in the bank. Considering how much we value money in our society, it's curious that we don't just tell others our bank balance in casual conversation. Instead of breaking this social taboo, we give cues to each other on our cash scores through what we wear, where we go on vacation, where our kids go to school, etc. Instead of shouting out our literal score, we signal it using status objects. In this point-model, signals tell the score, exact numbers don't.

Video game score

A much more overt score is that in almost any video game. The score is always there, blinking at the corner of the screen, letting the player know how close or far he or she is from the next level, other players and ultimately winning the game. Few systems in real life keep the score as omnipresent as video games.

A social networking score

When Facebook was developed, there was nothing overt to indicate that the number of friends a user had served much function whatsoever. Similarly, the number of followers or mentions on Twitter was never explicitly pointed to as a designated "score." But they are. Most users, if asked, can name how many friends and followers they have on any given social network. What's more, they can probably name which of their friends has an unusually high number of friends or followers. An inventory of sorts is taken, simply because Facebook and Twitter placed the "score" prominently on the page.

Keeping score

Google Orkut was a social networking site created by a Google employee if the same name. Most social network users in the world may not have heard of it. But there is one place in the world that Orkut is the #1 social networking site—Brazil.

While designing the site, Google put in place a simple leaderboard listing the number of people per country who had signed up for Orkut. When Brazil entered the top group on that leaderboard something unexpected started to happen—Brazilians began to host spontaneous Orkut sign up parties. The goal was to take over the number one spot held by the United States. Thus, Orkut became the country's number one social networking site, handily besting Facebook, all because of a score and an innocuous leaderboard.

Composite metrics

Any metric has the power to become a type of score. Sometimes it is better to create a composite metric in order to convey complex data in a simple form. A FICO score, for example, is an amalgam of a whole series of different pieces of information—from average monthly credit card payments to amassed debt over a lifetime. We could, of course, show different scores for each vector we wanted to measure. But by summarizing the complexity of creditworthiness in a single number, anyone from a prospective landlord to a bank clerk can derive meaning from it without needing a PhD in Economics.

Weight Watchers similarly creates a metric that calls on a series of numbers—from body mass index to weight to daily calorie intake in order to follow the progress of their users as they seek a healthier life style. In the same vein, Klout tells you and others how

you rank in influence and importance on Twitter. It doesn't take into account just a single vector, but rather a series of numbers that are amalgamated into one.

Point systems

In gamification, we can leverage one of five point designs to form the foundation of our experience. In some cases your point system will be overt, direct and highly motivational. In some designs you'll use four different kinds of points to achieve your objectives. In still others, points will take a back seat to other mechanics, doing their duties in the background as the designer's workhorse.

Whatever you end up choosing, you'll need to get a good handle on point system basics and options. Your points palette includes the following:

- Experience Points
- Redeemable
- Skill
- Karma
- Reputation

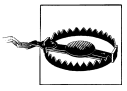
Experience points. Of the five kinds of point systems the most important are experience points, or XP. Unlike airline miles, XP does not serve as any type of currency within the system. They are, however, how you watch, rank and guide your player.

Everything a player does within the system will earn him XP—and, in general, XP never goes down and cannot be redeemed. By assigning XP to every activity in the system, you align your behavioral objectives with the player in a long-term way. In some systems XP can expire—say monthly or annually—to create goal loops. This pattern can be observed in the requalification periods used in frequent flyer programs—and expiry can serve the important purpose of resetting the game to level the playing field.

However, perhaps even more importantly, XP never maxes out. A player continues to earn them as long as she plays the game. That is the power of XP.

Redeemable points. The second point system is made up of redeemable points, or RP. RP, unlike XP fluctuates—It goes up and down. The expectation for most people is that these points are usable within the system to be exchanged for things. They are earned and cashed—like a bank balance or the frequent flyer miles we redeem for awards.

RP generally forms the foundation of the virtual economy, and is often given names like coins, bucks, cash, etc. Like any economy, you will need to monitor, manage and tweak the flows of capital to ensure everything runs smoothly, and to avoid massive inflation or deflation. In addition, redeemable points come with substantial issues from a legal and regulatory standpoint.



For more on the legal issues in redeemable point systems, visit GamificationU.com and complete the challenges.

Skill points. The third point system is called a skill point system. Skill points are assigned to specific activities within the game and are tangential to both XP and RP. They are a bonus set of points that allow a player to gain experience/reward for activities alongside the core.

By assigning skill points to an activity, we direct the player to complete some key alternate tasks and sub-goals. Classic examples of skill points are found in dungeons and dragons and other similar games here you have different skills, like magic and power, and a different score for each. In the non-game context, you might assign a set of varied skill points on a photo sharing website; players may earn some points for the quality of their photos and others for the quality of comments. Depending on the circumstances, it might be worthwhile to keep them separate.

Karma points. Karma points are a unique system that rarely appear in classic games. The sole purpose of karma is to give points away. That is, players gain no benefit from keeping their karma points, only from sharing.

UserVoice is a good example of this system. Using karma points, players vote for and against potential features they'd like to see built. If the player's vote wins and those features are built, he gets karma points back to allocate to new activities, and so forth. The only purpose of points in the UserVoice model is to give them away.

Reputation points. Finally, reputation points make up the most complex point system. Anytime a system requires trust between two or more parties that can't be explicitly guaranteed or managed by you, a reputation system is key. Its purpose is to act as a proxy for trust.

The reason why reputation systems are generally more complex lies in both how they are designed and used. In general, they must incorporate a wide range of activities in order to be meaningful—and the extant design must consider incentives and unintended consequences. Moreover, because they are a proxy for trust, players will certainly attempt to game the system. Integrity and consistency will be paramount.

How to Use Point Systems

To begin with, it's imperative to string an XP system around your gamified system. It informs you and your players about the activities that are more important.

Redeemable points, on the other hand, should be used when you want to create a virtual economy. Virtual economies are most valuable when you are looking to incentivize broad behaviors, large communities and/or leverage economics to drive behavior. However, they also have unique challenges, such as legal and regulatory issues that are complex and rapidly changing.

Another challenge of redeemable points is the issue of perception that comes with them. If you announce a great redeemable point system, the first thing a player will do is go and see what they can get. If what is offered feels neither meaningful nor realistic, you might lose the player altogether from the system. In other words, if a player looks at the redemption opportunity and thinks, "Yeah right. I'm not going to win a free car for watching a video," then that is one player who might not believe it is worth his while to stay within the system. Similarly, another might see that she could obtain free pizza for her points and think that she don't like or want free pizza. In both those cases, you might be at risk for losing players.

Reputation points are complex but often necessary in a system. The biggest problem with them, however, is that they are easy to "game." TripAdvisor is a website that shares customer reviews about travel worldwide. The site's success means that it is reportedly responsible for 30% of all hotel bookings. Therefore, hotels have a vested interest in not only seeing that they are favorably reviewed, but also that other hotels are not. Although no official statistics exist, a cursory review of TripAdvisor will immediately reveal a host of clearly "chaff" reviews.

Similarly, Yelp, a site that allows users to review local restaurants and entertainment venues, runs into a similar set of problems. To date, the only real way to determine which reviews are real and which are gamed is to read a lot of them. Neither of these sites have implemented a reputation system that is scaled to the level of complexity or value being transacted in the system.

Virtual economies

The power of a virtual economy is that it allows a designer to bring a lot of money in control how it goes out. Any macroeconomics major might recall a few communist and socialist countries that have operated under that very premise. In Cuba, a traveler can exchange any other currency for a convertible Cuban peso, but it can be nearly impossible to change it back.

That's how most economies are designed in the virtual world as well. For example, Zynga's Farmville has a completely one-way valve. A player can only put money in, and there are no real world rewards to redeem for; it all stays in the game.

In 2010, Zynga launched a well-known promotion with 7-11 and the Slurpee. Logically you might think that a player could exchange Farmville credits to pay for a Slurpee in store. However, the promotion was that for every Slurpee purchased, a player got free Farmville credits. The value of the virtual economy, in this case was greater than that of the real-world reward to those players.

Virtual economies and secondary markets. Secondary markets have been, by and large a bane to the designers of the MMOGS in which they have emerged. They were tolerated because their designers hadn't taken them into consideration in the first place. In new gamified designs however, the options for secondary markets are frequently greatly

reduced. Today's designers seek to control as many aspects of the virtual economy as possible—and secondary markets are in opposition to that objective.

Currency Denominations

The perception of the value of virtual currency can be closely tied to the currency that players are in. For example, the equivalent of $1 in the US is 1,000 Korean Won (they both buy approximately a soft drink from a street vendor). So when denominating a virtual currency for Korea, it's worthwhile to multiply the underlying numbers by 1,000 over the US. New social games automatically renumber all player views based on the player's country of origin, denominating everything in US$ behind the scenes. The parallels to the real world economy are not incidental.

Dual economy. A well functioning virtual economy will control player demand with minimum complexity. It allows for a certain level of fluidity in game play. If a designer creates a promotion without the help of a virtual economy, new and cumbersome explanations must be disclosed each time. For example, "Tell three friends about us and get a scratch off card that gets you 20% off this or 30% off that and if you tell four friends, we'll throw in a basketball and at ten friends..."

But in a virtual economy nothing more need be done to explain the promotion than to tell the player, "Tell three friends and get two hundred points." Two hundred points don't have to be explained. The player already knows what that will get him. Therefore marketing is optimized.

In fact, Farmville demonstrates this very well in what is called a dual economy. It has created two currencies within the system of the game: cash and coins. Conveniently, US dollars convert into both. Each is used for different kinds of items within the game.

Dual currencies are better for players who are already familiar with the nuances of the game. They are not necessarily suited for novices. So in the beginning of game dual economy designs are unimportant. They can reveal, however, how fine a control a designer can have over player behavior and demand if they have created a well-functioning virtual economy. Just by pulling these macroeconomic levers, you can guide your player through the system.

For example, a dual currency can enable you to set wildly different values on items within the economy while also controlling the inbound monetary supply. In the real world, this has been tried by so-called two-tier monetary systems, such as in Cuba or China, where they try to keep things cheap for locals and expensive for foreigners. While this approach has rarely worked in the long term in the real world, it's gaining prominence in virtual economies—but with the complexity caveat mentioned above.

EXERCISE

Assigned Point Values

When doing this exercise, don't consider how the points will be used or whether or not they are redemptive. Instead think about the relative value of each one of these actions. Which is worth more and by what percent based on your businesses' goals and objectives?

Keep in mind: In major social games there is a funnel—For every one hundred people that like something, ten of them convert and for every ten conversions a business earns $30. So, every "like" is worth thirty cents. And take note. This is a process that will change when it is put it into action. But in the beginning you must stake out a point value for every action in play.

Example: Assigned point values

Perhaps you chose as your top five social actions explore, comment, join, recommend and express and assigned values to them as follows:

Explore: 1 point
Comment: 2 points
Join: 4 points
Express: 4 points
Recommend: 20 points

You might reason that exploring the site is the least beneficial of the five actions to the objectives of the company, even if it is important to the gamified experience of the player. Therefore you assign to it the smallest value. Player comments, on the other hand, are worth double since it might lend quite a bit of value to your community at large.

Joining in general is an often under appreciated action. However, it is quite a hurdle to get players to sign up for your system. So in fact, giving it four times the weight afforded "explore" is actually the right idea. There is great value in having a player's email address and name—not to mention the new level of value she has attached to your system when she join.

By weighting "express" as four times more valuable than exploring you are making the statement that the social benefits of a player proselytizing on behalf of your system is a deeply important action—and perhaps even fundamental to your system. Obviously, recommend is among most extreme forms of player viral expression, so it is naturally weighted more heavily.

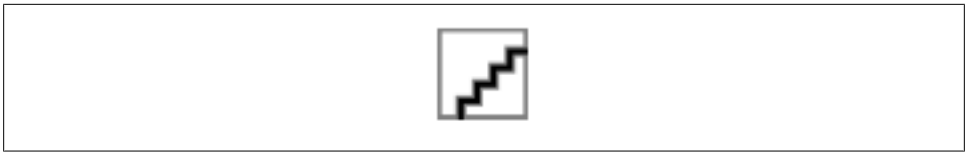


Figure 3-1. Level complexity

Levels

In most games, levels are the markers of progress—though they are not as exclusive in this role as they once were. In the arcade game Miss Pac Man, for example, levels are clearly expressed by the color of the ghosts, the layout of the board and the kind of fruit that loops around the board throughout the game. Of course, designers of gamified experiences aren't going to use traditional levels like those found in video games, but understanding them can add a powerful tool to your design. Levels serve as a marker for the player to know where they stand in a gaming experience overtime.

Level design

In Miss Pac Man, a player knows instantly that the level has changed because, in addition to being told directly and seeing changed colors, the game has gotten more difficult and the prizes for rewarded behaviors increase in value. Meanwhile, as the Miss Pac Man avatar moves at the same pace, the ghosts move faster, and the safety time zone delineations are shorter.

In game design, level difficulty is not linear. In other words, level one is not half as complex as level two, which is not half as complex as level three, and so on. Instead, difficulty increases in a curvilinear form. In the example of Miss Pac Man, an expert player knows that after level three, the ghosts slow down again and the safety time zone delineations increase. The board might continue to harbor more complexities, but like most level design, difficulty increases exponentially through each level and then decreases over time.

Complexity from one level to the next in the game Angry Birds has proven extremely engaging. Using powerfully designed levels the player passes almost seamlessly from one to the next, gaining confidence and experience. However, at one of the much higher levels in the game—level 21 in the first board, the player encounters a level that is decidedly more complicated than the one before it. It is, in fact, so difficult that there is only one sequence of events that will get the player through. It is the first time in the game that a player is likely to notice that he has passed to a new and more difficult level.

Some might consider the move by the designers of Angry Birds controversial. Inevitably players who find the challenge too much will drop out of the game. But on the other side of the argument, those who pass the level are more likely to feel like they've achieved something special and have become a part of a rarified group. Clearing the level will unlock the next board—so it's a major achievement—and one that bedeviled your authors for quite a while.

Progression of difficulty. The average length of the Nintendo arcade game, Donkey Kong, lasts less than 1 minute. The reason being that the first level of Donkey Kong was incredibly hard. Realistically, an arcade in the '80s would have a vested interest in their players losing faster thereby inserting additional quarters sooner.

In today's gaming systems, we are interested in longer, stickier games. Therefore today's designs start at the very simplest levels and move progressively toward the complex.

In PopCap's iPhone game Plants vs. Zombies, a player moves from one level to the next with the difficulty of the game increasing with each new level. A quick look at the game's first board and tenth board illustrates visually just how much more difficult the game becomes as the player moves forward. The board grows visibly crowded with characters and obstacles.

In some systems levels might define the complexity—or the leading element—of the game, or else they might serve as a passive marker to give more depth and complexity to your system.

Either way the best design tips for levels are to make them logical or easy for the player to understand, extensible so that you can add levels as needed beyond the initial "boss level," and flexible. Finally, the levels should be testable and refinable. Level balancing is just as complex as building the game in the first place and should be tested and retested even as the players are in the game.

Enduring leveling systems.

American Express has built an impressive level system using the demonstrability of the credit card itself. In fact, most Americans, if polled, could probably guess what you were talking about if you simply named off for them the colors of the Amex rainbow of credit cards: Green, Gold, Platinum and Black.

What is funny about that list is that, while green, gold and platinum all connote money and precious metals, the fact that they ended on "black" seems a surprising switch. Everyone knows very clearly that gold is more valuable than silver. By using "black" as their top tier, American Express changed the meaning of black. Now "black" as a top tier is as common as gold and platinum.

Precious Metals

How do we know gold is more valuable than silver and bronze? Unless you regularly monitor the precious metals market, odds are you were taught that fact by other leveling systems in your life, like the Olympics—that make use of precious metals in their level design.

University levels are similarly stamped with a clear ranking both within the confines of the institutions as well as for the population at large—a bachelor's, masters, and a PhD or doctorate, use language to clearly indicate the levels of mastery at the university level. The Military and the Boy Scouts have arguably two of the most perfected leveling systems of any institution. From badges and medals, to titles like General and Eagle Scout, even the uniforms indicate who resides where in the levels of the "game."

Progress bar. Progress bars have begun appearing all over the Internet. In most incarnations they tell a player in the form of percentages how close he is to completing all the necessary sign up information. Principally they are used to encourage new players to add personal information to a site or to create a deeper core experience. Progress bars work hand in hand with levels, serving as a progress guide for a player. Rather than naming completed levels, they work on percentages. Note: the best progress bars never reach 100%.

EXERCISE

Using a Metaphor

Like Amex or the Boyscouts, creatively describe the proposed levels for your gamified experience. Without using precious metals or gems, imagine what would be an interesting leveling system for your system.

Examples: Using a metaphor

Say you are developing a gamified experience for a company that sells women's shoes. You decide that it makes sense to title your levels based on candy to invoke both playfulness and color. The names you choose from the lowest level to the highest include —peppermint, cherry cordial, marshmallow, chocolate and truffle

While these levels certainly sound like they would be appealing to your demographic, sometimes the problem with a literal system is that people lose track of where they are, confusing for example, chocolate with truffle. Another thing to avoid is accruing a list of levels that come off as "cutesy", unless cute is in the honest voice of your player.

Finding your voice

One experience relayed from a prospective client was telling. The client made a type of financial marketplace software connecting investors and deals. Their clients, investment bankers, have already assigned a value to animals like bulls, bears and pigs. So when developing a gamified experience for that demographic they used them in the

leveling system. However, imagine attempting to engage a fifty-year-old investment banker with the image of an adorable pig—it might not work. But that's exactly what they did—to deleterious effect. Suffice it to say that they had to remove the too-cute-for-words characters and replace them with something more demure.

So while, sometimes a designer might be drawn to using words that relate to a specific community—in the case of a bull, a bear and a pig, or even in the case of those chocolates and peppermints—maybe the level should not be represented by a literal or cartoon representation. Perhaps it would be more effective to use a color to represent the image, or even a depiction of the word itself, depending on your audience. It's all about knowing your player.

Leaderboards

The purpose of a leaderboard is to make simple comparisons. Unsurprisingly, most people don't need any explanation when they encounter a leaderboard. By default, we see an ordered list with a score beside each name and we understand that we are looking at a ranking system.

In any 80's arcade, a novice who might approach a Galaga or Moon Patrol cabinet would find on the screen a list of high scorers—most of which trailed enough zeros to render a potential player dumbfounded. Talk about a terrible disincentive to play the game! Even when the number scale is completely meaningless or opaque the player still feels that four million points is a lot and probably not easily attainable.

The no-disincentive leaderboard

The leaderboard of today has seen some radical redesign since the heyday of pinball machines and quarter arcades. In the era of Facebook and the social graph, leaderboards are mostly tools for creating social incentive, rather than disincentive.

They accomplish this simply by taking the player and putting him right in the middle. It doesn't matter where he falls in ranking order—if he is at #81 or 200,000—the player will see himself right in the middle of the leaderboard. Below him, he will see friends who are on his tail and above him he will see exactly how close he is to the next best score. And he will know exactly what he has to do to beat it.

If the player, however, is actually in the top ten or top twenty, then the leaderboard should reflect this directly. In the case of these players, the leaderboard should show them their literal ranking since that is likely to be meaningful for them.

The infinite leaderboard

In an arcade, barring a daily deletion of the high scores, there are not too many ways to allow every player to exist on a given game's leaderboard forever. At some point her score will be beaten and she will fall off—or she will hit a number and sit there for

weeks until someone finally beats it. In today's world, there are ways to control leaderboards such that no player ever falls off or gets stuck.

Doodle Jump, a popular iPhone application allows its players to see the leaderboard sliced in various ways. A player can look at it locally, socially and globally. A local view shows him where he ranks compared to others in the system in his immediate vicinity. Socially, he can see how he ranks among his friends and followers in the game. A global view allows the same thing within the system as a whole.

There is no reason players can't slice and dice their leaderboard however they want. In fact, tracking the leaderboard behavior of a player will also inform the designer about her players. For example, a player with a deep interest in leaderboard positioning is likely to be a more competitive player and can be guided accordingly.

Leaderboards can also be displayed with a limited available view for the player. In a game with millions of players, this can be an important tool. Flight Control—the air traffic control game mentioned in chapter one—has a leaderboard that displays other players playing at the same level, ranked by distance and time.

Other popular social leaderboards include Klout, which, for example, ranked fans of the Sacramento Kings basketball team by their Klout score—showing that user's social power on Twitter. Also, Yelp's mobile app ranks a user's top weekly check-ins. The leaderboard can be cut by friends or by royalty—the high scorers in the game, similar to mayors in Foursquare.

Privacy and leaderboards

Sometimes creating a leaderboard isn't as obvious as it seems. In the event that the items being compared are sensitive or difficult to quantify, leaderboards present a unique challenge. Since the role of the leaderboard is to publicly compare, how does one compare information best kept private?

For example, a gym has a vested interest in helping its users achieve more healthful lifestyles or meet their fitness goals. Therefore, asking a novice to walk in, step on a scale and have his weight compared with other gym members is probably going to lose that gym quite a few perspective members. Not only is sharing a person's weight publicly a potentially shame-inducing experience for some people, not everyone joining your gym is there to lose weight. Some people want to train for a marathon while other want to relax, or even gain weight.

The first thing that might be clear is that more than one leaderboard might be necessary to meet the goals of the gym. For a novice player, a leaderboard that lists her attendance might be a great introduction to the system. Runners might want to be a part of a leaderboard where they race other gym members. And while body builders might even want to share their weight and watch themselves grow publicly, people seeking to lose weight might be less inclined to play if a public humiliation is part of the game. Further,

there is the potential that any of these leaderboards might induce unhealthy results if players push themselves to win.

Creating leaderboards using sensitive or private information is challenging but not impossible. Abstracted point systems can ensure that each player maintains a program that is healthful for them while sizing up their accomplishments in a public leaderboard. Ultimately, the designer will have to keep their goals in mind and maintain an awareness of their overall objectives—and take some responsibility for the power of the leaderboard.

In some cases, with the truly competitive, a straight leaderboard can be a powerful tool for motivation. For most explorers, socializers and many achievers, it can be both positive and negative. Consider your players' motivations and make your leaderboard social: it's a win-win proposition.

CHAPTER 4

Game Mechanics: Designing for Engagement – Part II

Badges

Although it's easy to forget, Foursquare did not invent badges. They've been around for a long time and are distinctively omnipresent in our world. On the back of most cars a small string of numbers and letters tell everyone driving on the road behind it everything they need to know about the car—and even a little bit about its driver. That badge signals what kind of engine is under the hood, what kind of price tag was on the car at the dealership, and therefore what kind of driver is behind that wheel. Carmakers prominently display those badges knowing full well that car owners take them very seriously.

In addition to signaling status, people want badges for all kinds of reasons. For many people, collecting is a powerful drive. Other players enjoy the sudden rush of surprise or pleasure when an unexpected badge shows up in a gamified system. A well-designed, visually valuable badge can also be compelling for purely aesthetic reasons.

For game designers, badges are an excellent way to encourage social promotion of their products and services. They also demark the meeting of goals and the steady progress of play within the system.

Boy Scouts and the Military

Two of the most enduring badging systems ever developed are those of the Boy Scouts and the military.

In the Boy Scouts, badges serve as a tangible point system. If a scout collects a certain number of badges, he is automatically elevated to the next level in the system. In the military, badges are a public display of accomplishment. In both cases, they serve as a reward for the completion of an action that the institution deems important and worthy.

The power of badges to motivate in the military and scouts is so powerful that the core systems have remained intact for hundreds of years. Have you ever participated in a comparably engaging badge system?

Effective Badging

In some designs, badges can replace levels as effective progress markers. For example, Foursquare uses check-in counting badges to demarcate levels instead of having a separate leveling system. As a result of the social, collectible and visual nature of badges, an increasing number of gamified systems are following in Foursquare's lead.

Another common conceit that we call "badgenfreude" suggests that an endless parade of boring and pointless badges have rendered all badges vapid at best and patronizing at worst, leaving many of us believing that badges suck.

However, just because a designer hasn't seen a well-executed badging system, doesn't mean they don't exist. And, to be clear, Foursquare isn't the only badging system in the virtual world with some level of credibility. But since it is a success-story in a place where Badgenfreude is the norm, for our purposes, let's look at where it succeeds:

Foursquare principally uses badges to represent progress for its players and to create a sense of delight or surprise. One of the most interesting things about their system is that it doles out those badges with a fair amount of seeming randomness. How and when Foursquare will badge its players is not always transparent. In other words, the players usually don't know what badge comes next. This decision is a controversial one since the lack of specific goals might frustrate a more competitive player. However, Foursquare's strength is in actualizing pleasant surprises by catching its players off guard with badges.

Controversy and Badging

When the "Douchebag Badge" was introduced to Foursquare, players had a tremendously split reaction to it. If you check in to 3 locations tagged as being popular with a certain kind of clientele, you'll earn the badge. Initially, as it was impossible to know where those locations were, almost all of the badges were earned unexpectedly. And while some people love it and some people don't, this badge clearly invoked the "voice" and spirit of Foursquare.

In both Farmville, ribbons serve as a badging system for players in the game. However, unlike the badges in Foursquare, these ribbons have different tiers for each badging objective, and act in close concert with challenges. For example, a player can earn four ribbons for being a good Samaritan in Farmville that are given at various levels of social promotion. In effect, she can earn the same badge four times with increasing degrees of achievement. So, while this system is decidedly more complicated than Foursquare's

"check in to win" system, it also shows the challenges more clearly to the player. In this way the player knows what has to be done to move on in the game.

Combining Surprise with Predictability

In most cases, designers find that a good balance between Foursquare's surprise badges and Farmville's predictable ones will meet the psychological objectives of players. But don't forget collecting and visual appeal!

Badge Examples: the good, the bad and the ugly

The Huffington Post is a good example of a poor badging system. For one, the proposed badges don't do much to convey the intended progress to mastery. Neither do they offer much in the way of visual value. They are also lacking a topical or social angle. Fundamentally, it just doesn't seem to be a very compelling system—more like a throwaway or afterthought. Though as with most gamification, the HuffPo has likely seen small improvements with minimal effort, there is much more that could be done.

GetGlue, on the other hand, badges players for promoting media products socially. Therefore, most of its products come with their own badge system. For example, the Fox hit TV show Glee has its own badging subset, as does Modern Family and The Office. Instead of merely badging for all TV consumption alone, GetGlue badges players for each individual property as well as their overall progress.

Similarly, a recent GetGlue promotion surrounded the rebirth of the television classic, Dr. Who. The site offered a limited edition badge for people who checked in for the first six episodes of the new series while it aired in real time. For anyone who is not a fan of the show, there is very little incentive to this promotion. But if you are—what could be more compelling than a limited edition badge designed for a show you love. And once that badge is gone, it is gone. GetGlue successfully leveraged scarcity and socialization to create great badges.

EXERCISE

Badge Design

Challenge yourself to design your own badge. What does a badge look like with regard to your products or services?

Keep in mind: Consumers respond to good design.

Customization

Customization can come in many forms. A game designer might leave it to his players to dress up and trick out their avatars or virtual worlds. In most gamified systems (in

comparison to games) the demand for 3D avatars is fairly low. However, even a simple player headshot and screen name can be considered an avatar, and provides opportunity for customization.

So while you probably don't need to worry much about customizing beach or forest scenes in your experience, allowing a player to select the color of her background or the font that writes out her name can actually add value to your experience. It is also a great way for her to spend her virtual currency and earn experience.

Customization is Commitment

One of the simplest examples of customization as commitment can be found in Twitter. All players have the option of changing their photos and background images. Although there is no cost or gamified experience around it, it's obvious when someone has not bothered to customize his view that he is exhibiting a low level of commitment to twitter (and may even be a newbie). Most designers believe that customization is a powerful tool for creating commitment and engagement.

The Tyranny of Choice

But be warned, customization has a dark side. From Barry Schwartz's 2004 work titled The Tyranny of Choice, wee learn that people are most satisfied when choice increases from zero to one. Satisfaction then tends to increase proportionately to the number of options. However, he cautions, only to a point. When there are too many choices, satisfaction drops precipitously. In brief, enough choice is good. Too much choice is bad.

His research divides people into two groups: *maximizers* and *satisficers*. (*Satisfice* is a portmanteau of satisfy + suffice). When looking to buy a new car, maximizers, according to Schwartz, would have to see every car option available on the market before they could make a decision. Satisficers on the other hand, define minimum criteria for choice, e.g. that they have $16,000 to spend on a two-door coupe. When they find the first car that meets those specifications, they simply buy it.

The research (backed up by personal observation) clearly shows that satisficers are generally happier people. So, when it comes to gamified options, it isn't good to reveal the entire complexity of the system up front. Give the player just enough choice to engage her without overwhelming her.

Customization with Apple

Apple has a very specific worldview on this—their customization of choice basically comes down to laser engraving and white vs. black. Those are pretty much the sum of the choices Apple gives to its users when they are buying their products. There is some

customization when it comes to choosing a computer—but unlike Dell.com where you find page after page of customizing options—the process is much more directed.

Leveraging Customization

Customization is not a panacea. Throwing a ton of options at people will not make them happy. And if you as the designer are not careful, they can be overwhelmed by a tyranny of choice. However, if you offer players a small number of well-placed customization choices in the flow of your experience, you can get them to demonstrate commitment and educate them on your process. Especially if you choose to use a virtual currency, having customization options will be a key way to redeem currency without hard-dollar cost.

Onboarding

Onboarding is the act of bringing a novice into your system. It is a carefully calculated way of thinking about how someone goes from zero to five miles per hour without crashing his car. Although there is a standard web design way of looking at onboarding (throw a huge number of options at a player to make sure he does something, anything) the game view is very, very different.

Lessons from the casual games market have shown that the first minute to any player is the most important. Most of a player's decisions are made in that first minute of engaging with a system. Just as Malcolm Gladwell describes in his book, Blink, we are trained to "thin-slice" all kinds of situations and people. That is, our animal brains are wired to make snap decisions about friend or foe, and ask questions later. Casual and social game designers understand this incredibly well. They think about players entering a funnel, and try to maximize the value and effect of that first minute. Train, engage—but don't overwhelm.

The Order of a Player's First Minute

The first minute a player spends with your system is not the time to explain anything to anyone. Instead, allow the player to experience the site. For example, the first time a player arrives at the dating site, HOT or NOT, she is asked to rate an attractive person with a simple yes or no question—Is this person hot or not? Immediately, the player is experiencing the core behavior of the site and with that, she is drawn in.

A fundamental mistake of many systems is to ask a player to register before getting to, for example, rate the first attractive person and experience the site. But for a player, there is nothing to compel him to want to give out personal information to a service he doesn't yet know. Conversely the agenda of the designer to get valuable information and data from a person might seem overt and therefore off-putting to the player.

So the second thing a good system will offer in that very first minute is something of value. In the case of HOT or NOT, the player is offered the opportunity to view and rank more attractive people. Other sites might offer prizes, achievements (like badges) or virtual items. No matter what the offer is, it should have value for the player.

Then, and only then, ask her to register.

TMI: Too Much Information

It never fails to amaze when a website thinks it can throw a novice right into the expert, or even master level of a game. Also offensive are sites that attempt to mask this flaw by putting up a messy page of exposition attempting to explain the game in eight paragraphs or less. People online are busy. They have better things to do than read about a game they have yet to care about.

But sites continue to make this mistake. Gowalla.com, for example, introduces itself with a long and complicated written introduction instead of letting players experience the core check-in loop itself. Why? Facilitating a more interactive first experience, along with the rewards that are so carefully built into it, would definitely serve them better.

The truth is, almost nobody reads that booklet that comes with every new Monopoly game. If the game isn't taught by someone who already knows how (a speech that nearly always finishes with, "Just play. You'll figure it out"), one person in the group is likely to skim the instructions and share the abridged version with the other players.

So expecting a player to spend time parsing out all of those words, numbers and complex subheadings on a site like Google AdWords—which in its current incarnation has arguably one of the worst designs for novice users—is absurd. A site like that is designed for failure. There is no way for a first time user to win. Yet still, they have the gall to ask anyone who comes along to create an ad, pay money, and wait for it to drive traffic to their products and services.

Obviously, Google has had tremendous success with AdWords, its self-service ad platform, but the experience is a turn off that definitely depresses revenue. Novices fail, lack comparisons to gauge performance, and are given too many choices: "the Google content network? Really?" To their credit, Google is aware of the issue and actively working on it.

Make Winners

Do not set up your novice player to fail on his first interaction with your game. Frontierville offers a very simple approach. Their very first screen features a cartoon character introducing himself:

"Hi, they call me Frontier Jack," he says. "Let's start by digging up the grass with the yellow arrow pointing at it."

The image on the screen shows Frontier Jack, a yellow arrow and a patch of grass. No one can mess this up. All a player has to be able to do is see a yellow arrow and click on it.

At the tutorial level, at level zero, there should be no choices. A player should be offered an action at which he cannot fail. Then he should be rewarded for successfully completing that action. (Even a "well done!" or a hearty "I agree," places your player squarely in a very seductive positive reinforcement loop.) This model, pioneered by social and casual games, has powerful implications for any kind of business, especially a gamified one.

In a nutshell, you want to offer players a clear path that follows the basic pattern:

Action
Reward
Action
Action
Reward
Join (register)
Invite friends

The stages can vary, as can the interval between them, but the basic pattern is enduring. Begin simply by asking players to take a no-risk action. Follow that with a reward, and continue the loop, slowly revealing the added complexity of your system, while also training players on how to achieve. Even better if you can learn something from the player from her behavior without having to explicitly ask.

Guiding Player Experience

Netflix offers an interesting model. While we would argue that they mistakenly ask their users to register immediately, once a user joins he is hit with a series of meaningful questions that allow the system to get to know him. Using multiple-choice questions the site instantly organizes the user's experience. By asking, for example, which do you like better, this movie or that one? They can glean all kinds of important data.

Asking the player meaningful questions truly informs his ongoing engagement. If your system is going to show players something they didn't know before, especially using some type of artificial intelligence, a training game can be most useful.

So begin by training your player by allowing him to engage in the core experience of the site. Next offer him a reward. Then slowly begin revealing the complexity of the game.

Remember: A player can't lose at Netflix or at HOT or NOT. Everyone playing wins.

EXERCISE

Design an A vs. B Quiz

Like those used in Netflix or at HOT or NOT, ask your canonical player a question as part of an onboarding experience. Allow the question to be fun yet informative for you. It can be expressed visually or with words.

Write two questions: An opening question and a follow up question dependent on the answer chosen by the player.

Example: Design an A vs. B Quiz

Take the example of a shopping site that attracts both buyers and sellers:

A. I like to get stuff.

or

B. I like to get rid of stuff.

Reward: "That's so cool, me too."

If the player answers "A" you might ask:

A. I really like exploring things.

or

B. I really like to be shown things.

In the preceding example there's a tacit acknowledgement that the better we are at assessing the needs and wants of our player the more likely we are to retain her. When your site is made up of a humongous pile of possible options behind the scenes—in order to avoid a tyranny of choice, questions like these can help narrow down the experience for your player from day one.

Challenges and Quests

Challenges and quests give players direction for what to do within the world of the gamified experience. Some people enter the game with no idea of its goals or fundamental drives. How much fun would a scavenger hunt be if you were told to just go and find some stuff? Nothing specific. Just anything. There isn't much challenge there. There is zero intrigue and an absolute lack of structure.

Even if a challenge isn't at the front and center of the experience, using challenges as an option somewhere in the body of the system can add depth and meaning for the player. Chocolatier is a game that has a great and satisfying set of challenges always a the ready.

The idea is to ensure there is always a challenge for players to take—ideally a few. They should be able to come into the experience and always have something interesting and substantial to accomplish or try that is on the path of your overall player experience. Some players will play challenge after challenge in sequence, trying to demolish as much of the game as possible. Others will just take one as needed to maintain interest. Your job is to craft interesting options and a large volume of them.

Hint: The player should not be given master level challenges as a novice. Different challenges for different levels are appropriate and fundamentally more successful.

Cooperative Quests

The most difficult type of quest to build is one that requires other players in order to complete them. Cooperative questing experiences, as they are known, are dependent on a community of players. In organizing a soccer game, the difficult part is not finding a pair of goal posts or a ball. It's getting 22 people to show up at a pitch at the same time and play.

In the beginning of the design process for a gamified system, it is better to design a game that is single player to that can evolve into cooperative play as the player gains mastery and more players join the game. In this vein, Gowalla offers a series of challenges and quests a player can follow any time she want to. It isn't simply that a player checks into a place because she is there, but that she can complete a challenge after she checks in. She can even announce her own quests and complete them for rewards. Like she can commit to walking 12 km for breast cancer. She checks in at one end and then checks in 12 km later. Her friends know about her achievement and the player takes pride in the accomplishment.

Obviously, cooperative designs are more socially powerful. Consider that the dark side of the soccer example we gave (a high minimum bar) also tends to increase the social/reward power of the experience (lots of people). If your community has a large number of people in it already, and their connections are somewhat extant, you should strongly consider designing something cooperative. You can also design for single player experiences in a group context, where players are playing alone, but their achievement rolls up to a group, or the results are shared/scored with a group. Sometimes even just having the rewards doled out in a group setting can be enough to trigger that response.

As an example of single player challenges in a group context, Jimmy Choo ran a successful promotion with Foursquare to announce their new casual shoe-wear line for women. Players who checked in at the right time could win one of 6 pairs of these not-yet-released shoes. There was an axis benefit: While the prize was valuable, status was the driver. How compelling for players that only 6 people worldwide were going to have those shoes!

EXERCISE

Create the First Challenge

Compose the first challenge after your player is onboarded.

Hint: Scavenger hunts, tag and exploration are good go-to options.

Social Engagement Loops

Social engagement loops, while not exclusive to games, borrow heavily from a viral loop design. A designer must not only see the way a player engages with the system but also how he leaves it and perhaps even more importantly, what brings him back again.

In a social engagement loop:

A motivating emotion leads to player re-engagement that leads to a social call to action which flows to visible progress and/or rewards which loops back around to a motivating emotion.

For players of Twitter the novice view of the engagement loop is as follows:

Motivating emotion = Connecting and Expressing
Player re-engagement = @Mentions
Social call to action = Tweets
Visible progress/reward = Followers

In summary, a novice player of Twitter will decide that she wants to connect and express what she thinks. Once she has done so, she may leave the system, but if someone mentions her in a tweet (known as an @mention), she will re-engage with the system. This then leads her to tweet back in response to her @mention. As a result, she gains followers because people on the site find what she has to say relevant or interesting. Thus, she is motivated yet again to connect and express herself.

Expert Players of Twitter

The game for a novice player always varies slightly from the game played at the expert level or beyond. So using the same engagement loop, we will illustrate how an expert player leaves and is brought back to the game:

Motivating emotion = Collecting and ranking
Player re-engagement = Tweets and retweets

Social call to action = Follows retweets
Visible progress/reward = Listing followers

In summary, an expert player will be motivated by how he is ranked in the system. He will be focused on how many followers he has and how that compares to other players in the game. Re-engagement, in addition to the @mentions, is going to also come from an interest in making other player's lists, leaderboards (in Klout, for example) and having their tweets retweeted. The social call to action has more depth for an expert player. As a novice he didn't necessarily understand what a retweet was, but now he does. Finally as his following and status in the game grows, there's a visible reward.

Sobering Thought:

Data indicates, that based on a series of flurry metrics about iPhone app usage, after 30 days, a free iPhone app loses 95% of players. Gamified engagement, especially in mobile apps, is critical to ensuring your network success.

The social engagement loop is important. As a designer it is vital that you are clear about what kind of player engagement you looking for and to hone what you need to make sure that your players come back each month.

EXERCISE

Create a Social Engagement Loop

Create a social engagement loop for your canonical player. Think of it as the first social loop in which they will find herself.

Example: Create a Social Engagement Loop

The example will take an educational mathematics website and look at a relevant engagement loop:

Motivating emotion = Exploration by parents and educators looking for tools
Player re-engagement = challenges set up by the system and other players
Social call to action = score each others problems
Visible progress/reward = accruing points for successfully completed challenges

A social engagement loop is important to design at every level of development of the system. No matter how detailed or even simplistic you want your system to be, a social engagement loop can help you generate viral growth. Sometimes the loop for the novice will look similar to the loop of the expert—the motivating emotion is the same as the social call to action. In fact, they don't have to be different.

In the Twitter example, "@mentions" and "followers" were present at both the novice and expert levels. If something is core to the design, its importance will reappear again

and again. Creating these loops allows deliberateness and focus on the things that get players involved, keep them engaged, and bring them back at every stage of development

Gaming the System

Do not be mistaken; if there is something of value in a system, the system will be gamed. This should not be a blocking thought at the early stages of design. It is merely a statement of fact. People attempt to exploit any system in which there is something they deem of value.

For example, before the financial crisis of 2008, had the head of the Fed been a game designer instead of an economist, there would have never been an assumption that a market could ever be self-policing. Instead, they would argue that there is no such thing. A system of self-policing is valuable but it doesn't solve all problems. In the real world, we employ actual police because we need them.

Having said that, there is no such thing as foolproof security. In any arena, it just doesn't exist. People will push the margin on everything they can. In the world of gambling, there is card counting for example—and a constant state of evolution between the cheater and the system designer.

Policing Your System

One basic way you can protect yourself is to create admin (administrative) or sysop (system operator) positions. The best part is that these roles can be played by volunteers and serve as a reward for success within your game. Power, as mentioned, is one of the most motivating and enduring rewards in any system.

A second way to protect your system is to write a great "terms of service" and apply it consistently in every sphere. In the lawsuit against the game Second Life, one of the items to emerge was that the creators hadn't stayed consistent with their terms of service. They wrote one thing but didn't stick to it. Ultimately, that can get you into trouble.

You should also seek to employ a system that allows you control over all transactions in a very finite way. For example, in almost all MMOGs, the system admins or the volunteer admins are all able to stop and roll back transactions at any time, without a court or tribunal. If someone is trying to exploit your system you have to be able to stop, give out the appropriate suspensions with time to investigate and then allowing players back. Give your admin and sysops the latitude to do that. Allow them to look for unusual behavior and things out of the norm and then make sure they are able to take immediate and decisive action.

Finally, don't over think the policing components of a new design. What you can do is iterate, repeat actions again and again, and make sure you pay attention to mistakes and successes so that as you lean, you grow.

Agile vs Gamification Design

Iteration is a core hallmark of agile design. In truth, agile and gamification have a lot in common. They both profess that any concept in a system requires repeated testing. All games should in fact be rife with testing loops. No gamified system should be built with a set-it-and-forget-it mentality. It doesn't work. Players level out, get bored, game the system, or leave it altogether. By avoiding iteration, the system is certain to end up where you don't want to be.

In an agile design, prioritization is similarly important, narrowing the designer's focus on a limited number of specific items. Agile design looks for the minimum viable product before launch—what a designer needs now, knowing they can change it later.

In gamified design, an XP system assigning a point value to everything your player does, is at minimum, the one absolute before the system can launch. The XP system must be able to report back about the players so that the designer can watch his engagement internally. Over time, this will transform any process. In part, it will highlight the system's top players. Players with the highest and most recent scores in a well-designed, well-balanced XP system are the most important players in the game.

Further, the more gamified your market is, the more gamification you will need in your system. If you are launching an airline, you probably need to employ a fairly evolved reward system from day one. If you are launching a community bank on the other hand, you probably have some months or even years before the competitive pressure really gets to you.

Empty Bar Problem: Foursquare

If your system suffers from 'empty bar problem,' in other words you need a community of players to make your system interesting—you need gamification. It's the best way to get over the empty bar hump.

Dennis Crowley and Naveen Selvadurai, started with a mobile location-based social network called Dodgeball before founding Foursquare. Dodgeball was bought by Google and later shut down. When they could, Crowley and Selvadurai launched a new version of Dodgeball on the iPhone but this time, they added game mechanics. Gamification unpacked something for Foursquare that had been missing: The problem, it seemed, was that it wasn't enough to have people say, "I'm here!" if there was no one around to reply, "Oh wow, me too!"

Think about how many people would need to take part in a system before there is any probability that more than one player would randomly be in the same place at the same time! Dodgeball counted on serendipity. It depended upon the natural occurrence of two people to be in the same place at the same time both of whom would think to check in with Dodgeball.

Game designers leave nothing to chance. The entire experience of a player is in some way contrived, or at least optimized to maximize the odds of success. Although it might seem serendipitous, it almost never is.

In a pick up game of online poker there is nothing coincidental about finding people already playing in a room when it's 2AM in San Francisco. In the early days of those games, companies would hire people to fill the rooms so that when a player came along, there were always people ready to play. No matter your level in the game, designers made sure that a player of your ilk matched to you. If you were an expert, so was the paid player. Just like intrinsic motivation, don't depend on chance to drive your market. These are things gamification exploits immediately.

If you have an empty bar problem, if you depend on a mass community of players to function, gamification is important because it gets you over the hump.

Groupon

In the early days of the social shopping site Groupon, the company deployed something called the SOS mechanic. Basically, in order to get half off a spa service 25 people needed to buy it. So a motivated user could send out an SOS to all of her friends in order to drive traffic to the deal. It was a challenge. The user could "win" a treatment for half off if she could motivate 24 other people to "win" one too.

Now the mechanic isn't as important because there are enough people in most of Groupon's major markets to make their deals viable without it. But the SOS mechanic helped them to overcome the empty bar problem in the early days.

Sometimes these gamification tools are a means to an end and sometimes they are the sweetener to trigger great long-term engagement from a player. Either way however, they should be deployed with clear business objectives, a testing process that makes sense, and an eye on the long term.

Dashboards

Dashboards tell a designers what is happening in their economy, no matter how it is designed. It doesn't matter what kind of point system you have, because its job is to unearth correlations and anomalies among players. As an example, Zynga designers figured out that Farmville players will spend $35 in a month but that after $35 they will be much more likely to leave the game. So prompting players to buy more cash after they spent $35 in a month was a bad idea. This was discovered by analyzing dashboard data.

Zynga tweaked the experience so that instead of another ask for money, the game asked the player to complete a challenge, invite a friend, etc. in order to advance in the game. It is a small change, but if it's enough to move the needle a couple of percent and reduce churn, it's worth it.

Your dashboard ultimately protects your system. It can lead you to fresh and unexpected opportunities. Once you become familiar with the data points of your dashboard you can watch your revenue and understand your pipeline. Especially if you are using a redeemable points system, you will need to know the value of those points and monitor inflation. You will use your dashboard to track referrals, churn, and returns, until eventually you can monitor player sentiment.

By then, you will be well on your way to maintaining a meaningful gamified experience for your players.

CHAPTER 5

How to Make Forums More Fun

Predating even the foundations of the common Internet, forum software has been a basic component of networked interactions for decades. Though it's had ups and downs, forums continue to be an essential part of the interactive experience online, and are probably already on your roadmap (if not already implemented). Most forums already make use of user status to distinguish between junior and senior members, but a more explicit gamification strategy can help make standard forms more fun.

In this chapter, we'll show you how to gamify a basic forums site. As a starting point, we'll use an easy-to-set-up open source Ruby on Rails project, called Altered Beast (*http://github.com/courtenay/altered_beast*). Altered Beast is a well-coded, no-frills forums application that includes basic user functionality that can we can extend, such as account creation and forums participation. In short, it's a perfect place to start for a gamification tutorial.

Planning a Gamification Makeover

Before jumping into the code, let's spend a little time thinking through how we can apply game mechanics to our forums software.

All forums, regardless of theme, work best if they become online communities. Regular users tend to establish personas on the site and friendships are often made. Regular users check in often—sometimes dozens of time per day—to read new information, follow up on posts and start new conversations. New users stumble on the forums and must be quickly pulled in by interesting content or drift away.

The quality of the forums relies a lot on the quality of the users and their sense of belonging to something important. Popular forum sites take on a life of their own, with unique jargon, side conversations and enforcement of social norms. While most forum sites offer the same basic features—creating an identity, posting topic threads, commenting on previous topics and discovering new content—the specific implementation of these features has a big effect on how the community is formed. Turning a simple forum site into a vibrant, online community is a big project.

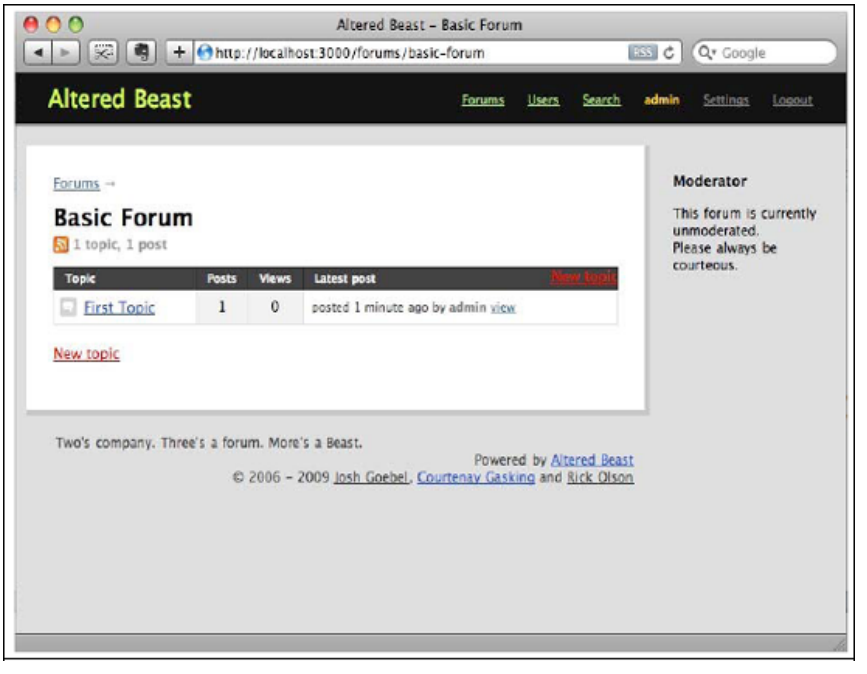


Figure 5-1. Altered Beast forums before a gamification makeover.

It requires the designer's effort and dedication but also a substantial investment of time and energy from users. Beyond the hurdles of creating an account and verifying identity, forums only work if users follow and participate in conversations. We can use game mechanics to turn users into players, by helping invest their identity with meaning (rank, status) and to provide incentives for taking actions that will endear them to the community. Because this principal challenge is psycho-social, the power of game mechanics are well-aligned with player motivations and interests, and we must focus our design on shaping their behaviors in appropriate ways. If we've done our job well, these incentives will ultimately satisfy the player, her community, and our company.

Our goals in this process should be to make using the site more fun, to incentivize players to act in the community's interest, to avoid incentivizing negative behaviors and provide a deeper sense of status and achievement to players who participate more fully in the site.

Points & Rules

Let's start by defining some rules for earning points. Points are a simple tool to reward behavior and an excellent base upon which to build a gamified design. The first step is to define the basic activities that we want players to take.

Table 5-1. Point awards for player activities. This defines the primary activities that we want to reward. These should be easy activities that encourage players to engage with the site.

Points	Activity	Purpose
10	Create an account	An early bonus to keep the player from starting with zero points.
10	Post a reply	Encourage the player to participate.
30	Start a new topic	Bonus for starting a new thread should be higher than simply replying.
10	Log in once per day	Reward players for coming back to the site frequently.

Designing Levels

Entire volumes have been written about level design. Finding the right balance between challenge and achievability is what makes level design a proprietary discipline with many skilled practitioners. As such, balancing levels is outside the scope of this book, but we encourage you to review the bibliography for a list of books with excellent insights on level design.

Example 5-1 defines a list of primary player activities on a forum site. Except for the create account bonus, these are all core activities that we want to encourage the player to repeat regularly, such as logging in to check threads and participating in conversations. Together, they form the basic 'gameplay' of our forums. We've included a one-time signup bonus so that their first experience with the scoring system is positive and engaging. When the player first notices their score, it will not be 'losing' score of zero. It will also encourage curious or competitive players to begin exploring the points system: they are likely to begin hunting for information about where these ten points came from and how to earn more.

Agile Reactions to Gaming the System

This simple point scheme is designed around the behaviors that we (as the site's owner) believe will engender the right outcome. However, this design also creates an incentive for players to game the system (by posting chaff comments to quickly earn points, for example). For now, let's ignore the risks of malfeasance and address those (known and unknown) issues after we get our basic site live. In this way, we can preserve an agile approach to game mechanic development and focus on the first order of business: getting community built around our site.

Some counter measures are described throughout this chapter, such as in section 5.2.5. We encourage you to consult them as guidelines for reigning in unwanted behaviors.

Levels

To increase a player's competitive instinct, we can implement a series of levels that confer rank as players become more active. A good and straightforward way to award

levels in a new gamified design is to base them on point thresholds. As players earn points, they move up an incremental series of levels. To invest the level with meaning —a sense of rank in the system—the levels should be named in a way that indicates status and the level should display along with the player's headshot or avatar throughout the boards. More complex designs, such as awarding levels based on consecutive actions or achievements, are also possible. But starting with a points-based system makes it easier to tune your new game economy after you open your doors. If you find that only 1% of your players make it to level two, you can easily lower the threshold without a major recode.

For our forums makeover, we'll define a few levels based on earning specific point thresholds. We need to be careful to define the levels sequentially, with appropriate minimum scores, so that the levels are indeed consecutive. Let's define a series of level scores that gradually get harder to achieve: 0, 50, 100, 200, 350, 600, 1000 and 2000. We would like the player to quickly earn some status, but have to work harder and harder to get to the top. This will give all players a chance to feel like they're accomplishing something, but ensure that the most dedicated are not bored by rising up and out too quickly.

Table 5-2. Levels. Levels are a way to both reflect player status and encourage competitive behaviors

Level	Name	Points Required
1	Fresh Meat	0
2	New Around Here	50
3	Wallflower	100
4	Learning the Ropes	200
5	Something to Say	350
6	Know-it-all	600
7	Expert	1000
8	Guru	2000

Choosing Level Names

In general, you will probably want to name these levels with names that are meaningful to your players, choosing fun names that convey both rank and the meaning of that rank. Many forum level naming schemes include ranks like Newbie, Expert, Genius and Founding Member. Take a look at your favorite forums for clues about what works and what doesn't in your industry.

Badges

Another way of explicitly nudging a player to action is to award badges for completing some tasks. Badges both reward player actions and provide an additional measure of

status on the site, since badges give players a way to show off their accomplishments. A well-designed badge system should let the player earn a few badges easily and then get progressively more challenging. For our forums makeover, we'll set up a few basic badges.

Table 5-3. Badges for player activities. Badges provide additional incentives to take actions, as well confer status on the player.

Points	Activity	Purpose
Newbie	Create an account.	An easy badge, just to get the player warmed up.
Chatterbox	Post 5 comments.	Award a new player who begins to participate in a significant way.
Icebreaker	Post 3 new topics	Encourage players who start new conversations.
Talk of the Town	Post a topic that gets 10 comments.	Reward a player for starting a particularly interesting conversation.

Think of badges as a way of rewarding the player for an extraordinary achievements or earning new skills. Depending on your site and your audience a well-designed badge system to can launch players in to a 'mission mode', where they go far and above the required effort just to earn a badge (and its concomitant bragging or laughing rights). Examples would be the "crunked" badge that Foursquare awards after a player checks into enough bars or Gowalla's "Hacker" pin for checking in at "25 technology start-ups". Badges can even be tied to special features or privileges that are unlocked of certain behaviors. An example would be to create a "Moderator" pin and give well-rated players the ability to delete comments in a forum. There are particularly powerful, since players are doubly incentivized: not only do they receive special powers, but also other players can see that from their pin. All well-designed badge system, appropriate to your site and players, can be a powerful motivator.

A system for tracking scores and levels

Now that we have a basic game design in hand—with points, levels and badges clearly defined—we can start to modify Altered Beast to make use of our design. We'll start by implementing a general framework to award and track points and levels. Once that's in place, we can add business logic to award points, based on our game design.

Creating a Level Model

Let's begin by adding a few basic models and methods to award player points. To do this, we'll need to define a levels model to keep track of the available levels. We'll also need to modify the player model to keep track of points and levels.

The level model defines hierarchical ranks, which players achieve by earning points. While there are lots of ways to assign levels, in our design we choose to award levels by cumulative points earned, since that makes it easier to adjust without recoding our

system. Once we have more data on how our players are progressing through the levels, we can adjust the points required to balance how hard it is to move through the levels.

To get started, we'll define an ActiveRecord model migration to create a levels table and populate it with the levels we defined in Example 5-2

Example 5-1. Creating a migration to define levels

```
class CreateLevels < ActiveRecord::Migration
 def self.up
 create_table :levels do |t|
 t.integer :number
 t.string :display_name
 t.integer :required_score,  :default => 0
 t.timestamps
 end
 Level.create(:number => 1, :display_name => "Fresh Meat",
   :required_score => 0)
 Level.create(:number => 2, :display_name => "New Around Here",
   :required_score => 50)
 Level.create(:number => 3, :display_name => "Wallflower",
   :required_score => 100)
 Level.create(:number => 4, :display_name => "Learning the Ropes",
   :required_score => 200)
 Level.create(:number => 5, :display_name => "Something to Say",
   :required_score => 350)
 Level.create(:number => 6, :display_name => "Know-it-all",
   :required_score => 600)
 Level.create(:number => 7, :display_name => "Expert",
   :required_score => 1000)
 Level.create(:number => 8, :display_name => "Guru",
   :required_score => 2000)
 end
 def self.down
 drop_table :levels
 end
end
```

This creates a table that we can use to set levels, which will give players a way to earn rank within the community. As a player earns points, we want them to "level up" at defined intervals, increasing their rank. Each level has a number, a display name and minimum score required to achieve the level. To make this lookup easy and consistent, we'll create a convenience method to find the correct level for a given score. We'll add this method to the `level` model:

```
    def self.find_level_for_score(score)
     Level.find(:first, :conditions => [ "required_score <= ?", score], :order =>
    "required_score DESC")
    end
```

The lookup method takes a score and returns the highest level that qualifies for the score. The method will work since we defined our levels sequentially with appropriate

minimum scores. As long as there is a level with a required score of zero, this method will never return a `nil` value.

Adding Scores and Levels to the User Model

Now that we have a model to define levels, we're ready to modify the existing player model to track the player's current score and level. We'll start with an ActiveRecord migration to add these fields to the user table.

```
class AddScoreAndLevelToUsers < ActiveRecord::Migration
 def self.up
 add_column :users, :score, :integer, :default => 0
 add_column :users, :level_id, :integer
 end
 def self.down
 remove_column :users, :score
 remove_column :users, :level_id
 end
end
```

This gives us a place to track the player's score and to assign a current level. We set the default score to zero, so that we don't have to worry about dealing nil scores in our arithmetic.

Next, we'll define our model relationships. Since we've already defined a `level` table, with an id column and since we added the foreign key `level_id` to the user table, Ruby on Rails makes it easy it define our table relationships. To do this, we'll one like of code both the `User` class and the `Level` class:

```
class User
 belongs_to :level
 ...
class Level < ActiveRecord::Base
 has_many :users
 ...
```

This describes the "`User` has many `Levels`" relationship, so that we can access attributes in a very readable way, by writing things like `user.level.display_name.`

Now we have a `score` and a `level_id` attribute defined on the `user` model. This would be a sufficient framework to allow us to jump right into coding business logic, but there is one problem: we have an attribute to track a player's score, but we have no way to log the activities that generated the player's score. This makes it hard to audit how a player achieved a given score, and limits our ability to explain to players how they the earned their points. Let's add one more model, to log events that generated points, so that we have a history of how a player achieved their current score. It will come in especially handy if we need to deduct points from a player's score.

Creating an Events Model

An `event` model will give us a way to keep a log of each time the player earns or loses points, so that we can track their point history. We'll start by adding a migration to create a table to store the event data.

```
class CreateEvents < ActiveRecord::Migration
 def self.up
 create_table :events do |t|
 t.integer :user_id
 t.string :text
 t.integer :points
 t.timestamps
 end
 end
 def self.down
 drop_table :events
 end
end
```

The `event` model will store a point value and a text description of why the transaction occurred. In the model class, we need to define the "belongs to user" relationship. And for convenience, we'll also define a default sort order for events, so that recent events will show up first in any listing:

```
class Event < ActiveRecord::Base
 belongs_to :user
 default_scope :order => 'created_at DESC'
end
```

That takes care of the `event` model. To take advantage of Ruby on Rails' ActiveRecord code for simple lookups, we need to define the "each user has many events" relationship in the user model:

```
class User
 has_many :events, :dependent => :destroy
 ...
```

The only things we're missing now are methods to award points to the player and update their score.

Extending the User Model to Scores and Levels

Whenever we add points to the player's score, we'll check to see if the player has achieved a new level. To do this, we'll add a convenience method to add points and some private lookup methods to set the player's score and levels.

Example 5-2. Extending the user model to add points and set levels

```
class User
 belongs_to :level
 has_many :events, :dependent => :destroy
```

```
def add_points(new_points, event_string)   ❶
update_score_and_level(new_points)
log_event(new_points, event_string)
end
private
def update_score_and_level(new_points)   ❷
new_score = self.score += new_points
self.update_attribute(:score, new_score)

new_level = Level.find_level_for_score(new_score)
if new_level &&
(!self.level || new_level.number > self.level.number)
self.update_attribute(:level_id, new_level.id)
end
end
def log_event(points, text)    ❸
events.create(:points => points, :text => text)
end
end
```

❶ The public method `add_points` is a convenience method that handles awarding points, updating the player's level and logging the event.

❷ The `update_score_and_level` method ensures that we update a player's score and level safely. It first increments and updates the score. We use the `update_attribute` method to quickly save the score to the database without triggering any model callbacks. Next, the method looks up the level for the new score. If the player does not yet have a level—a new user—then the new level is saved. If the player already has a level, then the level is updated only if the new level is of a higher rank.

❸ The `log_event` event method handles creating an event to log each change to player's score.

With this basic framework in place, we have all the code we need to start implementing some game logic. Since `user.add_points`, handles player score updates, checking for a new level and logging the event, it will be very easy to focus on the game logic to award those points.

Awarding Points for Key Activities

With a basic points system in place, we can focus our attention on awarding points for gameplay behavior. In Table 5-1, we defined four conditions for earning points: signing up, replying to posts, creating posts and logging in once per day. Since it's very likely that you'll want to change the award values for activities as part of your game-tuning process, it's helpful to define award bonuses as constants in an initializer file.

```
SIGNUP_BONUS = 10
POST_BONUS  = 10
TOPIC_BONUS  = 30
LOGIN_BONUS  = 10
```

Using consistent constant names will make the bonus award code easier to read and modify.

Award a signup bonus

To award a signup bonus we'll first need a private method in the user model. We can use a Ruby on Rails before_create callback to run a method just after a new user is created. At the top of our user model, we can define the callback and a new method to run:

after_create :award_signup_bonus

This tells Rails to execute an `award_signup_bonus` method just after a new user is successfully created. We can reuse the `add_points` method we created earlier to award the bonus in a new private method.

```
def award_signup_bonus
 add_points(SIGNUP_BONUS, "Sign-up bonus!")
end
```

The new method will be called in a transaction, right after the user record has been successfully saved. It uses our `add_points` method to give the user 10 points, log the event and update their starting level.

Awarding bonus points for replying to posts

Awarding bonus points for replying to topics only takes a few lines of code as well. Altered Beast already tracks some statistics every time a new post is created, so we need only add a callback on the Post model to award points:

```
after_create :award_user_points
```

and then define a protected method to award points:

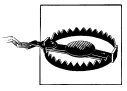
```
def award_user_points
 user.add_points(POST_BONUS, "You posted a reply.")
end
```

That was easy! Now each time a player creates a post, we'll award 10 points.

Preserving the integrity of your point model

There is small catch that you might notice while adding the callback. In the `post` model, just below the `after_create` callback is an `after_destroy` callback: it's possible for a moderator, or the player, to delete a comment in a thread.

If we do nothing else at this point, it will be possible for a player to earn points for comments that the moderator deems inappropriate—or even just by repeatedly posting and deleting one comment.



Counter Measures

Game mechanics are designed to engage a player's competitive instincts. As such, your design needs to anticipate that some players will try to "game the system". If you make it possible to cheat or earn points by repeatedly taking unhelpful actions without consequence, some players will definitely do it. Consider building some limits to the number of posts per day a player can make, or allowing others to flag posts which immediately removes the earned points. While the first version need not consider all these issues – and they are covered elsewhere in this book – it's good practice to anticipate that anything that can happen, will.

In the case of a forum with a community, the community will act as brake on inappropriate behavior, since truly engaged players will seek to avoid being banned or shunned by the moderator. While more sophisticated code to limit repeated or unhelpful posts—such as rating and flagging of comments—may be required in some forums, for our basic design we just want to ensure that a player is not rewarded for taking unhelpful actions.

In this case, we just need to add a callback to deduct points when a post is deleted, either by the player or a moderator. To make the code more explicit, we'll first create a `deduct_points` method on the user model:

```
def deduct_points(points_to_deduct, event_string)
 add_points(-points_to_deduct, event_string)
end
```

This method takes a positive point value, makes it negative and the then calls the `add_points` method. This will allow us to write a more readable callback on the post model:

```
after_destroy :update_cached_fields, :deduct_user_points
```

The `after_destroy` will call a new private method on the post model to deduct these points.

```
def deduct_user_points
 user.deduct_points(POST_BONUS, "Your post was deleted.")
end
```

The `after_destroy` callback method calls an explicit deduct method on the user model. Now, when a moderator or player deletes a post, the player's score will be reduced, which should limit the incentive to over-post, but still encourage a player to post relevant comments.

Awarding bonus points for starting new topics

When a player creates a new topic, we can award and deduct points in the same way that we did above for new posts, by adding new `after_create` and `after_destroy` callbacks to the post model.

```
after_create :create_initial_post, :award_user_points
after_destroy :update_cached_forum_and_user_counts, :deduct_user_points
...
def award_user_points
 user.add_points(TOPIC_BONUS, "You posted a topic.")
end
def deduct_user_points
 user.deduct_points(TOPIC_BONUS, "Your topic was deleted.")
end
```

Now, when a new post is created we add points to player's score. If the player or moderator later decides to delete the topic, the player's points will be withdrawn.

Awarding a login bonus

Awarding a login bonus is a powerful way to motivate players to come back to your site frequently. Before implementing the feature, you will need to think a little about your site design and how players log in. In our example, players will receive a bonus when they visit the site, up to once per day. If you want players to visit your site more often, you might want to change the interval to every n hours, but it's a good idea to limit this kind of bonus to avoid encouraging the player to log in and out repeatedly just to earn points.

Deciding whether or not the player should earn the bonus is very straightforward; the trick is deciding how often to check. You can imagine a timed interval award check line of code to look something like:

```
award_login_bonus if last_login_bonus_awarded_at > 1.day.ago
```

If the last bonus was awarded more than 1 day ago, the player should get a new bonus. The right place to call this line of code will depend on how your site handles sessions. If it uses sessions that expire relatively quickly, like a bank that requires the player to log in every hour, then you can trigger the bonus check each time the player logs in. As long as your sessions are shorter than your bonus limit, this could work well.

But what if your site lets players choose a "remember on this computer" option, so that they only log in periodically? If you allow players to stay logged in to your site for longer periods of time, then you will need to check `award_user_points` when a player views a page. Depending on how your site is architected, you can either insert this check as part of the session authentication or by adding a filter to your Application Controller. `The award_user_points` method would be called every time the player requests any page.

This could be a good option if you are already caching the user object for authentication or if you have no way to predict which views a player would visit. But you will need to be careful about performance, since the method will be called for every single page you serve.

In the case of a forums site, the behavior we fundamentally want to reward is reading the forums, so we can cheat a little bit and only the trigger the event when a player views a topic.

In fact, Altered Beast already uses this trick to track when a player was last seen online. Instead of updating the `user.last_seen_at` every time a player loads a page, Altered Beast only does it when they load a new topic, via the `Topics#show` method. Since this is the same behavior that we want to use as player reward, we can piggy back off the `user.seen!` method to grant our login bonus. If your site is structured in a similar way, you might want to use a similar design.

The first the thing we need is a place to store the time when the player last received a login bonus, so we'll start with a migration to add a new field to the database:

```
class AddLastLoginBonusTimestampToUser < ActiveRecord::Migration
 def self.up
 add_column :users, :last_login_bonus_awarded_at, :datetime
 end
 def self.down
 remove_column :users, :last_login_bonus_awarded_at
 end
end
```

This adds `a last_login_bonus_awarded_at` attribute to the user model, so we can store a timestamp when the player last earned the bonus. Next, we'll add a method to the user model to award the login bonus. We'll include the decision logic inside the method and also handle some errors:

```
def award_login_bonus!
    unless last_login_bonus_awarded_at \
     && last_login_bonus_awarded_at > 1.day.ago.utc

     add_points(POST_BONUS, "Daily login bonus!")
     write_attribute :last_login_bonus_awarded_at, Time.now.utc
     end
    end
```

Since new players will not have a `last_login_bonus_awarded_at`, we must first check for a nil. If there is `last_login_bonus_awarded_at` populated, then we check whether the date is greater than one day ago. Since we only care about relative time—how many hours have passed since the last award—we'll store the awarded at date in Universal Standard Time (UTC), to avoid having to worry about timezones.

If `last_login_bonus_awarded_at` is nil or older than one day ago, then we call our `user.add_points` method and update the database. We make use of the `write_attrib ute` method once again, since it performs a fast database update, without triggering all of our user model validations.

All that's left is to insert a call to this method in the `user.seen!` method:

```
    def seen!
     now = Time.now.utc
     self.class.update_all ['last_seen_at = ?', now], ['id = ?', id]
     write_attribute :last_seen_at, now award_login_bonus!
     award_login_bonus!
    end
```

Altered Beast uses the seen! method to update the player's `last_seen_at` date each time they view a topic. We'll check for our award bonus at the method, ensure it gets called frequently, but not on every call.

Now we've completed a basic set of rules for awarding player points and created a series of models to track the data. Next, we'll display some of this data, so that players can see how they are doing.

Badges

Now we have implemented code to award points as players "play" our site and increase their level as they earn points. The only thing left is to award badges for extra achievements. Back in Table 5-3, we defined four types of badges to be awarded for various accomplishments. To award those badges, we'll first need a Badge model to define the types of awards.

Example 5-3. Creating a badge model

```
class CreateBadges < ActiveRecord::Migration
 def self.up
 create_table :badges do |t|
  t.string :name
  t.string :display_name
  t.timestamps
 end

 Badge.create(:name => "newbie",  :display_name => "Newbie")
 Badge.create(:name => "chatterbox", :display_name => "Chatterbox")
 Badge.create(:name => "icebreaker", :display_name => "Icebreaker")
 Badge.create(:name => "talk_of_the_town", :display_name => "Talk of the Town")
 end
 def self.down
 drop_table :badges
 end
 end
```

The `badge` model functions much like our level model, defining a range of possible awards that a player can have. Unlike levels though, players can have many badges at one time. This sets up a many-to-many relationship between badges and users, so we need an intermediate model to track which badges a player has earned. We'll call this model Achievements.

Example 5-4. Creating an achievement model

```
class CreateAchievements< ActiveRecord::Migration
 def self.up
 create_table :achievements do |t|
 t.integer :user_id
 t.integer :badge_id
 t.timestamps
 end
```

```
 end
 def self.down
 drop_table :achievements
 end
end
```

With the achievement table in place, we'll need to add a few lines of code to the models, so that Rails understands the relationships between the Achievements, Badges and Users tables.

```
class Achievement < ActiveRecord::Base
 belongs_to :user
 belongs_to :badge
end
class Badge < ActiveRecord::Base
 has_many :achievements
 ...
class User
 belongs_to :level
 has_many :achievements
 ...
```

With these lines of code in place, Rails knows that each User can have many Achievements and that each Achievement has a badge. To make awarding badges easier, let's also add a convenience method to the User model.

```
def award_badge(name)
 badge = Badge.find_by_name(name)
 achievements.create(:badge => badge)
end
```

This will allow us to make a one-line, readable call like user.award_badge("icebreaker") to award a badge. The method will look up the correct badge, create a new achievement and log the event in the player's history. Now all we have to do is write the code to award the badges.

Awarding The First Badges

In Table 5-3, we defined a series of badges that we wanted to award. The first was called the Newbie badge and was awarded simply for showing up. We already defined an after_create method on our user model, so we can expand it to award this badge by adding one line of code to the award_signup_bonus method.

```
def award_signup_bonus
 add_points(SIGNUP_BONUS, "Sign-up bonus!")
 award_badge('newbie')
 end
```

This revised method will now award a new player their signup bonus. We'll then use the award_badge method to issue the first player badge.

Subsequent Badge Awards

The next badge, called the Chatterbox, is awarded when a player posts their 5th comment on a post. Since we've already created a callback on the Post model to award points each time a player creates a new post, we can simply expand this code to award the badge.

```
def award_user_points
 user.add_points(POST_BONUS, "You posted a reply.")
 user.award_badge('chatterbox') if user.reload.posts_count == 5
end
```

Here we've added a new line to the award_user_points method to award the Chatterbox badge on the player's fifth post. In most games, a player can lose points but not badges, so we won't code a method to revoke a player's badge if their post is deleted. However, we have created one problem with this new logic. If a player creates five posts, then deletes one, then creates a new fifth post, this award will grant him a second Chatterbox badge.

Usually a player can only earn one of each type of badge. Let's refactor our user#award_badge method slightly to enforce this rule for all badges.

```
def award_badge(name)
 badge = Badge.find_by_name(name)
 unless self.achievements.find_by_badge_id(badge.id)
 self.achievements.create(:badge => badge)
 end
end
```

The revised method will first check to see if the player already has this badge. It will only award a new achievement if they do not.

The third badge that we defined in Table 5-3 was the Icebreaker badge. This is similar to the Chatterbox badge, but is award when the player creates their third new topic. We can add a line to the existing topic#award_user_points method to award a Chatterbox badge when a player creates their third new topic.

```
def award_user_points
 user.add_points(TOPIC_BONUS, "You posted a topic.")
 user.award_badge('icebreaker') if user.topics.count == 3
end
```

This new line works almost exactly the same way as Chatterbox badge issuance. The only difference, aside from being called when a new topic is created, is that Altered Beast keeps a posts_count cache on the User model, but not a topic_count cache, so we have to call the count method on the player's topics in order to calculate when to award the Icebreaker badge.

The final badge that we need to implement is the Talk of the Town badge, which the player earns when one of their topics receives 10 posts. Since the trigger for this award will also occur when a new post is saved, we can further expand the post#award_user_points method to award it.

```
def award_user_points
 user.add_points(POST_BONUS, "You posted a reply.")
 user.award_badge('chatterbox') if user.reload.posts_count == 5

 if topic.posts.count(:conditions => ['user_id != ?', topic.user.id]) == 10
 topic.user.award_badge('talk_of_the_town')
 end
end
```

To avoid incentivizing players to post repeated comments on their own topics just to win this badge, we don't count a player's own replies when awarding this badge. The topic poster receives a Talk of Town Badge when their topic gets 10 replies from other players. We implemented this in a straightforward manner using ActiveRecord's count method with a condition to filter out the topic creator's posts. When that count equals ten, the topic creator receives a Talk of the Town badge.

Now we have implanted our entire gamification makeover design. Players earn points, move up to higher and higher levels and earn badges for extraordinary achievements. Taken together, these systems should motivate players to work at becoming high-ranking members of the Forums community. To unlock that behavior, though, they'll need to see their rewards—and be able to show them off to their friends. In the next section, we'll walk through displaying this information to a player.

Displaying player scores and levels in the site

In addition to our internal design, we've given players two powerful outward-facing tools to track their progress on the site: score and level. We also created an event model, to give players a way to see how they are earning points. If we want players to find this information meaningful and easily learn the rules of the community, then we need to prominently display this information. We'll do this by giving players a status view that contains all their information and also by adding their current score and level to the header, where they can see it all the time.

Scores and levels are also a way for players to display their status to others, so we should also tie their scores and ranks to their public identity on the site. We'll do this by trying to display their score and level whenever we display their headshot in the site, such as next to a forum post.

Adding a Player's Score & Level to the Sidebar

To keep the player aware of their point balance, we'll add a small score box that appears in the right-hand column when a player is logged in. This way the player will be able to see their score in every view.

We make this happen by adding some display code to the application layout template, application.html.erb.

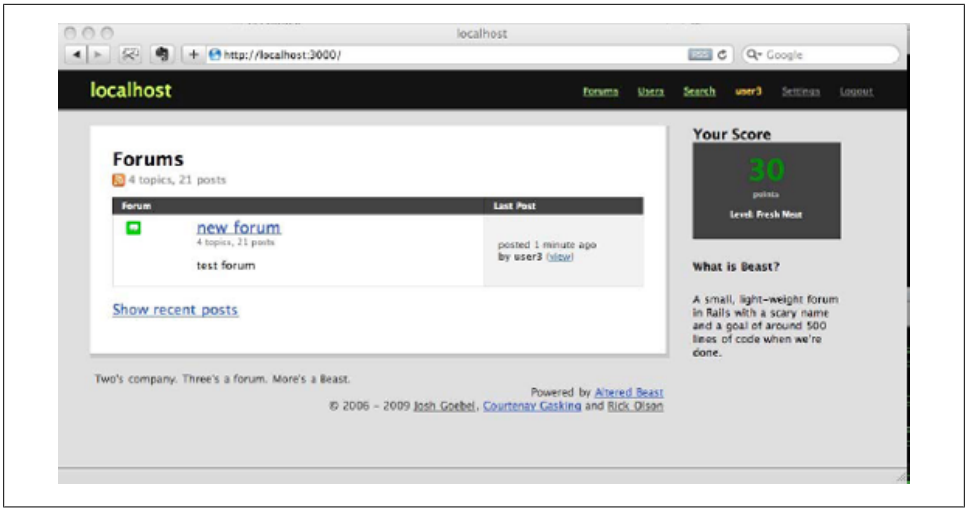


Figure 5-2. By adding the player's score sidebar, we keep their awareness on earning points.

```
<div id="right">
 <% if current_user %>
 <div id="notice">
 <strong>Your Score</strong>
 <div id="your_score">
  <h3 class="points"><%=current_user.score %></h3>
  <h4>points</h4>
  <h3 class="level" align="center">
  Level: <%=current_user.level.display_name %>
  </h3>
 </div>
 </div>
 <% end %>
 <%= yield :right %>
</div>
```

We inserted the code snippet in the existing div, just above the yield statement. This will ensure that the scorebox is always at the top of the right hand column, just above any custom notices that might appear in those views. We first check if the `current_user` parameter exists, so that our score box will only appear for logged-in players. The next lines format a display of the player's score and current level.

Adding a Player's Level to Topic Posts

Adding the player's current score to the right column as we did above ensures that the player stays aware of their point balance, helping them learn how to earn points. Now we'll add the player's level to their forum posts. This will associate their rank with their headshot, tying their status to their identity on the site.

Altered Beast already marks status by displaying a player's post count, so we can piggyback off that layout. In views/topics/show.html.erb there is snippet of HTML that writes out the player's post count under their avatar.

```
<span class="posts">
 <%= I18n.t 'txt.count_posts', :count => post.user.posts.size,
 :num => number_with_delimiter(post.user.posts.size) %>
</span>
```

The I18n.t makes use of the Rails' internationalization and pluralization features. In this case, if the player has 1 or 2 posts and their default language is set to English, it outputs a string such as "1 post" or "2 posts". We'll skip over internationalization for now, and replace the post count with the poster's current level:

```
<span class="level">
 <%=post.user.level.display_name %>
</span>
```

Now, every time a player posts to the forum, their current level will appear next to their headshot.

Adding a Basic Leaderboard

Now that we added a player's points and levels to the right column and began to print their current level next their headshot, we should be piquing their interest in points and levels. Their next question will probably be: "Who's winning?" We can answer this question by adding a basic leaderboard to the mix. This is one of the most powerful tools we have to motivate players to want to increase their scores.

Altered Beast already includes a rudimentary leaderboard: the players list. It lists all the players of the site and shows their post count:

This serves its purpose well as basic directory of users, but we can make it a more powerful tool by turning it into an explicit leaderboard. We'll do this by making a few changes:

- Make the first column a score
- Add a column to display the level as well
- Sort the table by score, so that the highest-scoring players appear at the top of the list

This gives a simple but usable leaderboard where the viewer can quickly see that the way to the top of list is by increasing their score:

Example 5-5. Turning the user#index view into a leaderboard

```
<table border="0" cellspacing="0" cellpadding="0" class="wide forums">
 <tr>
 <th>Score</th>          ❶
 <th class="la" width="100">
```

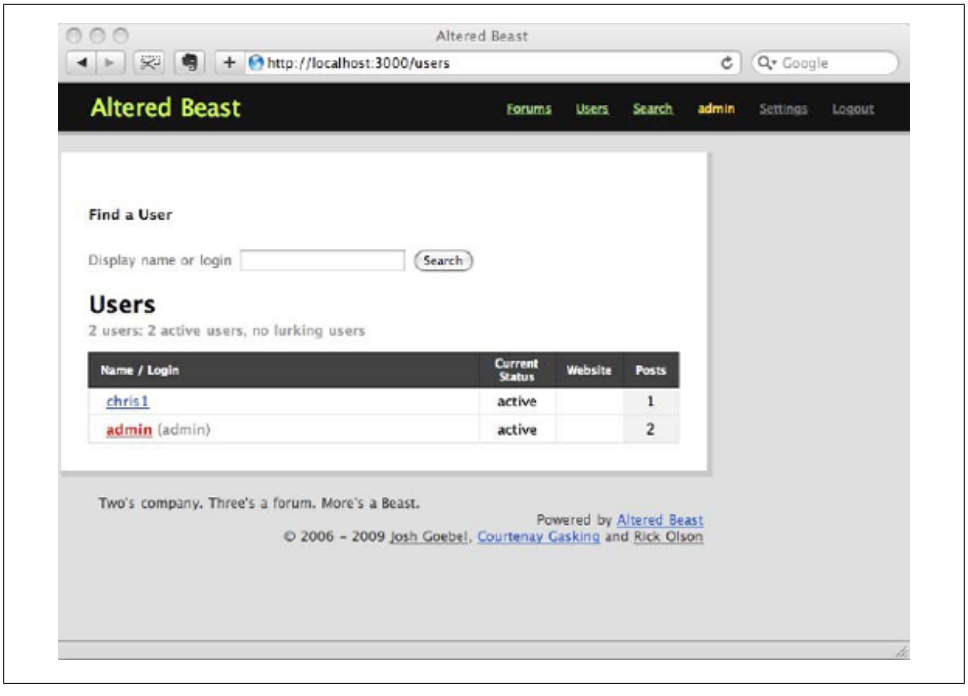


Figure 5-3. The Altered Beast users list already functions as a de facto leaderboard.

```
 <%= I18n.t 'txt.views_users.name_or_login', :default => 'Name / Login' %>
 </th>
 <% if logged_in? && current_user.admin? -%>
  <th>
 <%=I18n.t 'txt.views_users.current_status_title',
  :default => "Current Status" %>
 </th>
 <% end -%>
 <th width="200">Level</th>          ❷
 <th><%= I18n.t 'txt.views_users.posts_title', :default => 'Posts' %>  ❸
 </th>
 <th><%= I18n.t 'txt.views_users.website_title', :default => 'Website' %></th>
 </tr>
<% @users.each do |user|-%>
 <tr>
 <td class="ca inv"><strong><%= user.score %></strong></td>  ❹
 <td>
 <%=link_to h(user.display_name || user.login), user,
  :class => (user.admin? ? "admin" : nil) %>
 <span style="color:#666">
 <%=I18n.t('txt.views_users.admin_in_parens',
  :default => "(admin)") if user.admin? %>
 </span>
 </td>
 <% if logged_in? && current_user.admin? -%>
 <td><%= user.state %></td>
 <% end -%>
```

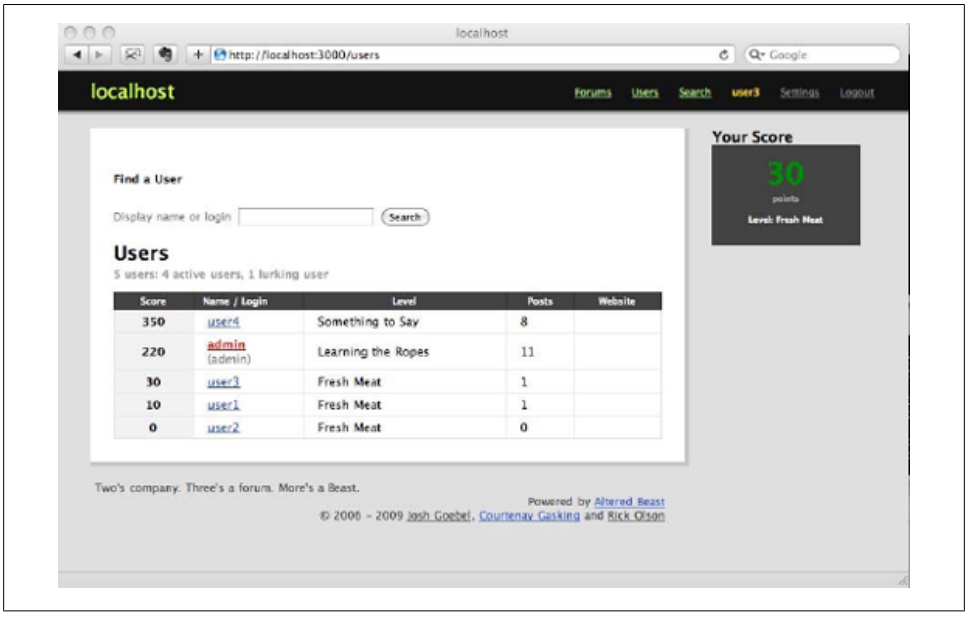


Figure 5-4. Sorting users by score converts the list into a competitive ranking.

```
 <td><%= user.level.display_name %></td>     ❺
 <td><%= user.posts.size %></td>
 <td class="la">
 <% unless user.website.blank? %>
 <%=sanitize link_to(user.website.gsub("http://",""),
  "http://" + user.website.gsub("http://","")) %>
 <% end %>
 </td>
 </tr>
<% end %>
</table>
```

❶ Make the first column "Score"

❷ Insert a "Level" Column

❸ Move the "Posts" column before "Website"

❹ Insert the player's score

❺ Insert the player's level

❶ Make Score the first column and use Altered Beast's class="la" tag to make the background grey, so the field will stand out.

❷ Insert the Level column header. Note that we're not using the I18n.t method in our columns for simplicity. I18n.t is a built-in Rails localization method that looks up

display strings depending on the player's locale settings. It's very useful, but complicates our example.

❸ Move the Posts column above the Website column, since it relates to Level.

❹ Within the @users.each loop, output the player's score in the first column. We used Altered Beast's class="ca inv" CSS to make the column background darker, so the score column stands out.

❺ Output the player's level column

Optimizing Leaderboard Output

This will output a simple leaderboard of all the players, but there is still one problem: the list is not sorted by score, so it's difficult to tell who is winning. We can solve that with a just a few lines of code. First, we'll need a named scope on the user model. This is a Rails convenience function that lets you define search criteria—in this case, a sort order—for a model and give it a clear name. We do this by adding a line to the user model:

```
named_scope :by_score, :order => 'score DESC'
```

What this tells Rails is: "when I specify a search by_score, order the results by score, in descending order." To use it for our leaderboard, we need to modify the line of code in the UsersController#index method that sends the list of players to the view. The current method loads a paginated array of all players for the current site:

```
@users = current_site.send(users_scope).paginate(:page => current_page)
```

In essence, the method asks the current site to return a list of players without specifying a sort order. The .paginate() method is a handy Rails' extension that handles breaking the list into manageable chunks that the view can use for display. To sort the array, we can make use of our new score-sorting scope:

```
@users = current_site.send(users_scope).by_score.paginate(:page => current_page)
```

We inserted the by_score scope at the end of the array method, just before paginating it. Now the line of code returns an array of players for the current site, sorted by score and paginated for the current view's page. That's all there is to it.

Easy Enhancements to the Leaderboard

Now we have a fully functional, if not very dynamic leaderboard. There are a number of ways that you could enhance the leaderboard to increase its appeal. A few simple extensions would be:

- Add headshots to clearly connect player identity and status
- Include an explicitly numbered rank, so that players could more easily see the top 10, 25, 50 and so on

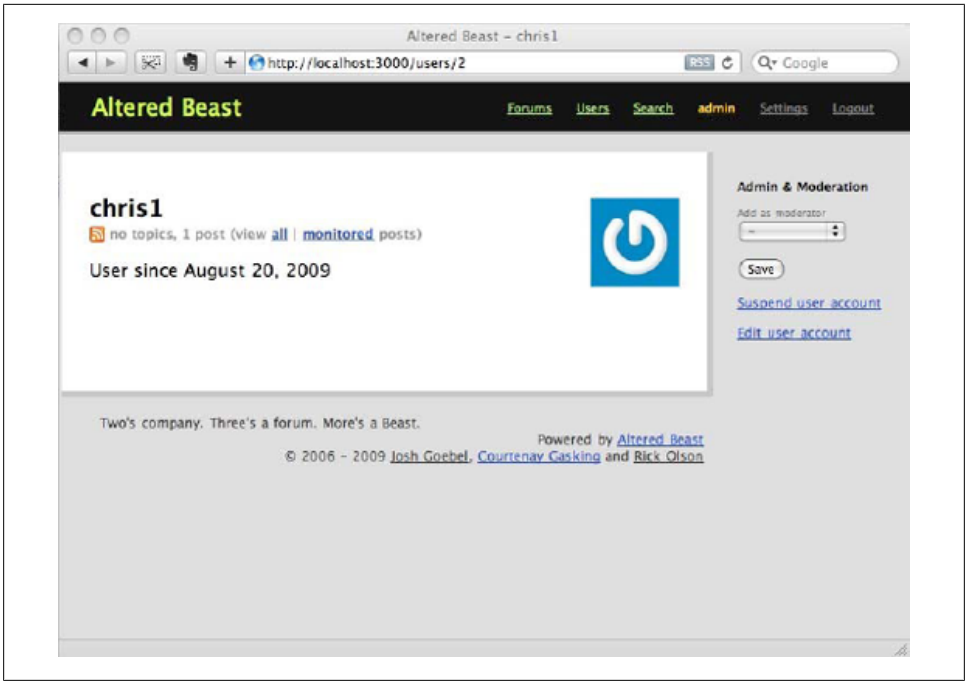


Figure 5-5. The player detail view before a gamification makeover.

- Add sorting, so that players can change the rank order by score, level or number of postings
- Rename the view to explicitly define it as a leaderboard in the top navigation

Next, we'll give each player a special place to show off all the points and status they've earned: a trophy case.

The Trophy Case

Winning something (such as a badge, achievement or level) is much less meaningful if your peers don't know of your accomplishment. The easiest way to convey this information is to create a Trophy Case or Trophy Room. This is the detailed display of the player's rank and achievements. In a more advanced design, with badges and virtual items, this would be the place to display the entire array of player achievements in one spot, accessible to all players.

Altered Beast already includes a simple player display, accessible from the leaderboard. If you click on a player's name, you'll see a page like this:

The current view is all business. It displays the player's name, headshot and posting count, but it's not very interesting beyond that. We can enhance it by adding the player's score, level and their recent history, so that it looks more like this:

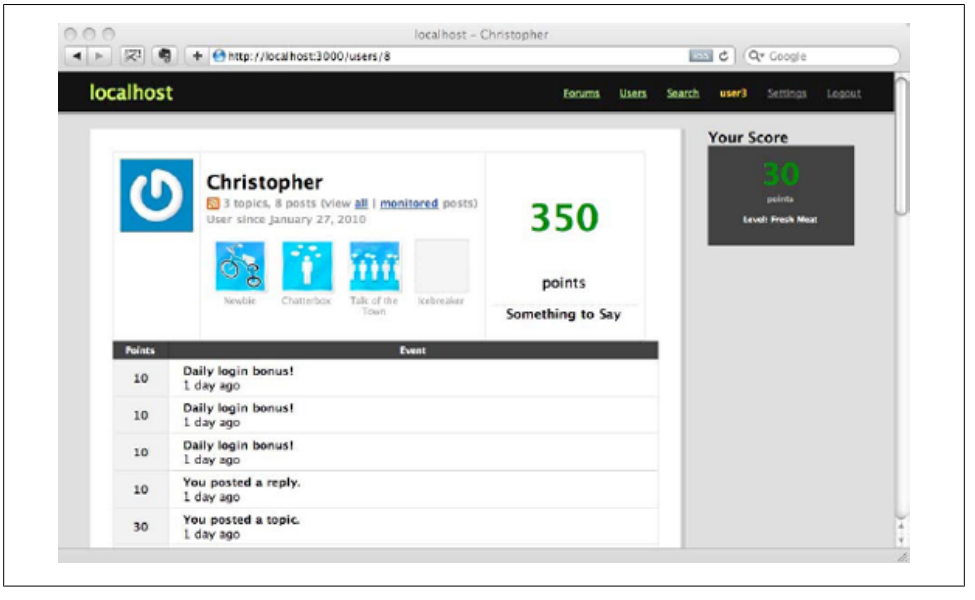


Figure 5-6. By prominently featuring points, level and badges, the user view becomes a trophy case.

To do this, we need to modify the user display view, located in views/users/show.html.erb. To turn the view into a simple trophy case, we'll include their achievements. Before we implement the view, however, there is one thing we could do to make it more powerful.

Listing the badges that a player has not yet earned

Since badges award special achievements, the can be powerful motivators for players. But in order to fully unlock that power, the players have to know which badges they can earn. One easy way to convey that information is to show a list of grayed-out badges that the player can still earn. We can add a convenience method to the User model to provide that list.

```
def unearned_badges
 if self.achievements.nil?
 unearned_badges = Badge.find(:all)
 else
 unearned_badges = []
 Badge.find(:all).each do |badge|
 unearned_badges << badge unless self.achievements.find_by_badge_id(badge.id)
 end
 end
 unearned_badges
end
```

The method first checks the player's achievements. If there aren't any, it will return all possible badges. Otherwise, the method loops through each possible badge and adds it to a hash if the player has not already earned. The method then return the hash of badges that player hasn't earned, which we can use in the trophy case to display all badges, earned and unearned.

Example 5-6. Adding a trophy case to the show user view

```
...
<table cellspacing="0" cellpadding="5" border="0">
 <tr>
 <td width="80" valign="top"><%= avatar_for @user, 80 %></td>
 <td valign="top">       ❶
 <h1><%=h @user.display_name %></h1>
 <p class="subtitle">
 <%= feed_icon_tag @user.display_name,
  user_posts_path(:user_id => @user, :format => :rss) %>
 <span>
  <%=I18n.t 'txt.count_topics', :count => @user.topics.size,
  :num => number_with_delimiter(@user.topics.size) %>,
  <%=I18n.t 'txt.count_posts', :count => @user.posts.size,
  :num => number_with_delimiter(@user.posts.size) %>
  (<%= I18n.t 'txt.view', :default => 'view' %>
  <%= link_to I18n.t('txt.all', :default => 'all'),
  user_posts_path(@user) %> |
  <%= link_to I18n.t('txt.monitored', :default => 'monitored'), "#" %>
  <%= I18n.t 'txt.posts', :default => 'posts' %>)
  <br/>
 </span>
 </p>
 <%= @user.bio_html %>
 <% unless @user.website.blank? -%>
 <p><strong><%= I18n.t 'txt.website', :default => 'Website' %></strong>
 <%= sanitize link_to(@user.website.gsub("http://",""),
  "http://" + @user.website.gsub("http://","")) %>
 <% end -%>
 <p class="subtitle">
 <%= I18n.t 'txt.user_since', :default => 'User since {{date}}',
  :date => @user.created_at.to_date.to_s(:long) %></p>
 <% @user.achievements.each do |achievement| %>    ❷
 <div class="badge">
  <%= image_tag("badges/#{achievement.badge.name}.jpg") %>
  <span><%= achievement.badge.display_name %></span>
 </div>
 <% end %>
 <% @user.unearned_badges.each do |badge| %>    ❸
 <div class="badge">
  <%= image_tag('badges/unearned.jpg') %>
  <span><%= badge.display_name %></span>
 </div>
 <% end %>
 </td>
 <td width="160" valign="top" align="center">
 <p id="user_score"><%=@user.score %></p>    ❹
```

```
 <p>points</p>
 <p id="user_level"><%=@user.level.display_name %></p>    ❺
 </td>
 </tr>
</table>
<table border="0" cellspacing="0" cellpadding="0" class="wide forums">
 <tr>
 <th width="40">Points</th>
 <th>Event</th>
 </tr>
 <% @user.events.each do |event|-%>       ❻
 <tr>
 <td class="ca inv"><strong><%= event.points %></strong></td>
 <td>
 <strong><%= event.text %></strong>
 <br/>
 <%= distance_of_time_in_words_to_now(event.created_at) %> ago
 </td>
 </tr>
 <% end %>
</table>
```

❶ Change alignment top for layout

❷ Loop through player's achievement badges

❸ Loop through unearned badges

❹ Display player's score

❺ Display player's current level

❻ Loop through events for display

The view code in Example 5-6 shows the changes we made to the existing user detail view. The view now loops through the player's badges and outputs the badge image and display name. Then it loops through the badges the player has not yet earned, retrieved from the User#unearned_badges method we created earlier. We also added a prominent display of the player's current points and score.

The last section sets up a table and then loops through the player events table to display the player's point history. That lays out all of the gameplay information that we want the players to see and should serve to focus their attention on trying to earn more badge, points and higher levels.

Summary

In this chapter, we've covered the basics for implementing gamification in a forums site. You've seen examples of how to implement the core backend elements of a game mechanics: tracking scores, measuring levels, awarding points. You also saw some examples of how to design and implement key awards to motivate players, such as signup, login and activity bonuses. In addition, we illustrated basic implementations of key

views: displaying an "always on" dashboard score, adding level to the player's site identity, implementing a leaderboard and displaying a trophy case.

Clearly, this is not the limit of game design for basic websites. The range of options that can be implemented to incentivize and direct player behavior are limited only by your imagination – but these basics form a critical foundation for moving to the next level: social networking.

As Facebook, MySpace, Loopt and other networks have proliferated, so too has the opportunity to leverage their open APIs and substantial traffic to bring new players into your site. By tying gamification to Social Networking, we have the opportunity to create a viral and community-oriented whole that is substantially more than the sum of its parts. Now, let's take a closer look at social networks and how create a bi-directional flow of players, data and fun between your site and your players' social graph.

CHAPTER 6

Badgeville: An Instant Gamification Platform

In the previous chapter, we walked through the implementation of a basic game-layer, coding out all the pieces by hand. Another approach, which can help you get started more quickly, is to use a Gamification platform like Badgeville.

Badgeville is a white label Social Rewards & Analytics Platform, with a suite of tools to help web site and mobile app developers leverage game mechanics and awards to identify, recognize, and reward users in an immersive social experience, along with a sophisticated analytics engine to help tune your implementation.

In this chapter, we walk you through an example Badgeville implementation to show you how to employ game mechanics and loyalty best practices to influence user behavior, grow their loyalty, and make your site or app successful.

Game On

Using game mechanics to build rewards and loyalty program can help turn visitors into fans and fans into advocates. The games need not be elaborate or overt. You can start with leader boards that feature top members of the week. You can use real time notification of award achievements displayed in a strategic location to provide an inviting hum of participation. The premise is simple, you define the important interactions on your site, and reward players when they participate or compete to perform those actions. When you're ready, you can add layers of more advanced reward-based interactions.

A well-implemented rewards program should help you convert your players from user IDs into people who actively help you understand what's good about your site or app, and what's not. Next generation analytics isn't based on page views, it's built using an engaging layer of people playing, enjoying, and sharing their experiences.

Critical Elements of an Online Rewards Experience

A social rewards program grows loyalty by making a significant, behavior changing connection with the user. Several key elements increase your chance of making this connection. An effective online rewards experience must use real-time feedback, leverage the social web, and be a vehicle for accomplishment.

The necessity of real time feedback is obvious: a user enjoys and appreciates instant gratification. More importantly, real time updates of status give the user confidence—she knows where she is and where's she's going. And when she can share achievement with her friends and get kudos from her network, the experience is even more powerful.

From a publisher perspective, social sharing is a powerful marketing tool. Understanding that 30% of Foursquare badges are shared into Facebook and that the average Facebook user has 150 friends, social sharing features are too useful to ignore.

An online rewards program can be a source of continued achievement. It offers a multitude of accomplishments of varying degrees of difficulty. While a user may not explicitly recognize micro achievements as a source of deep satisfaction, they are. The overwhelming success of apps like Farmville is an example of this.

Example Rewards Project from Design to Deploy

We'll walk you through the design and implementation of Skumo, a mock site for finding local businesses. Members review businesses and make comments on reviews.

Planning a Rewards Project

Gamifying a complete social rewards project requires an implementation plan that includes many tasks. For simplicity, we'll break it down into two domains: design and development. Here are a few key areas we'll cover:

- Design
- Selecting business objectives
- Defining behaviors
- Designing the games to achieve the objectives
- Defining levels
- Defining rewards
- Development
- Create your rewards program in the Publisher module
- Prepare methods to call the API
- Prepare methods to parse JSON data

Figure 6-1. The Skumo home page

- Register and track users
- Enable behavior tracking
- Create rewards data (leaderboards, profiles, activity)
- Create a rewards notification
- Integrate Badgeville with a commenting system

Selecting Business Objectives

In order to drive the most success, you begin the project by understanding its business objectives. What important business problem do you want to solve? A successful rewards program supports your ability to achieve core business objectives. Some common examples of appropriate objectives for a rewards program include:

- Increasing ad revenue
- Increasing sponsorship revenue
- Reducing content creation costs
- Reducing content moderation costs

Skumo's Objectives

Skumo's main objective is to increase ad and sponsorship revenue by providing useful content that grows membership. Skumo, the business, cannot afford to hire full-time writers and editors so it uses technology to drive a content creation and moderation process that elevates useful content. Skumo's challenge is universal for web publishers: how do you reduce costs while improving product quality? The answer is: tell your members what you need, make it fun for them to participate, and reward them for doing so.

Defining Desired Behaviors

Behaviors are the actions that users perform that help achieve your objectives. Typical examples of behavior include registering or reading articles. You should select a few behaviors to implement for an initial launch. Here are a few examples of behaviors that support common publisher objectives:

- Signing in
- Visiting a page
- Sharing a page
- Uploading a photo or video
- Buying a product
- Making a payment
- Voting or participating in a survey
- Providing feedback
- Participating in promotions

Skumo is successful to the degree that members add reviews, comments, and ratings, and share the content outside the site. Thus, Skumo promotes and tracks these behaviors:

- Adding, rating, reading reviews
- Adding, rating, reading comments
- Promoting content
- Becoming a member

These behaviors are discussed later in the chapter.

Design the Games

Once you understand the business objectives and the relevant behaviors, you can define games to achieve them. Games don't have to be complex, but they need to have either defined or implied winning conditions. For example, getting more points in a week than a friend, or achieving an enviable status.

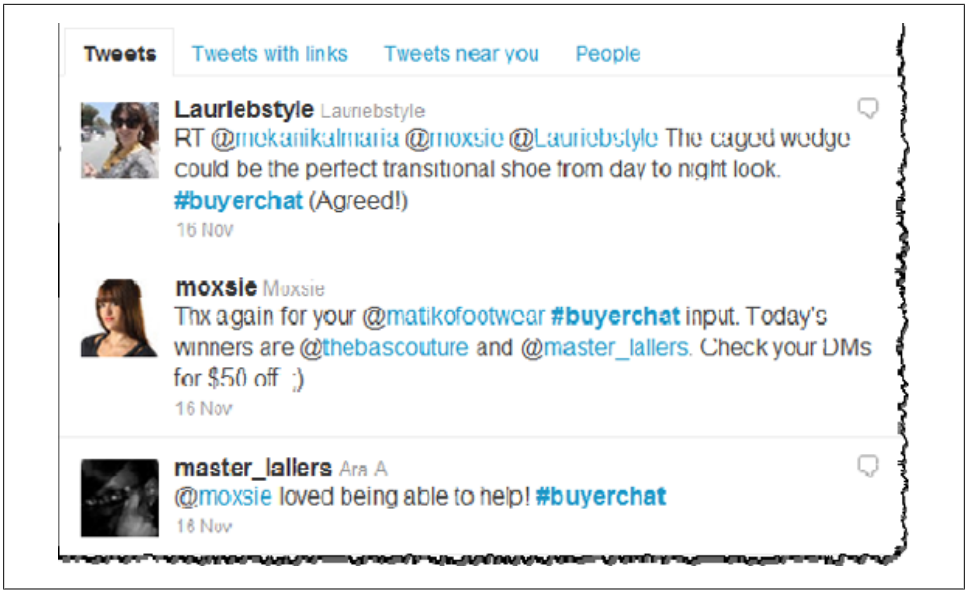


Figure 6-2. Moxsie's #buyerchat in Twitter

Figure 6-3. Personal, Friend, and Group motivators leveraged by game designers

To target specific types of users, game techniques can be paired with several types of personal motivators: personal, friends, and group.

Personal motivators resonate with an individual and satisfy core needs to achieve and be recognized for achieving. For some people accumulation can be a powerful motivator because it is a vehicle for self-expression. What you own may reflect who you are. For others, collecting rewards feels like being on a treasure hunt; you feel richer for the experience.

Friends provide an incredibly rich domain of motivators that can increase game design success. People value greatly the opinion of their friends, but they are also much more comfortable competing with friends and bragging about their own accomplishments with friends. Thus a rewards program that leverages reputation, comparison, competition and discovery among friends, will more likely succeed.

Similar to friend-based motivators, you can use group-based motivators to effectively engage your users. People like to stand out in a crowd and be known. They also enjoy comparing themselves with and competing against the anonymous crowd. Social games have used these motivators successfully for years. You should use them in your rewards programming design to increase participation and build a more loyal fan base.

Skumo Design. Based on our business objectives, reviewing, commenting, and rating, we identified several types of players: reviewers, commenters, and raters.

Reviewers respond to personal, friend, and group motivators. Reviewers want to become a site celebrity. Thus, they want more than just the sense of belonging; they want recognition.

Commenters are like reviewers, but may not have the time or ambition to craft useful reviews but they do want recognition.

Raters, at the other end of the spectrum, want to be useful and want to contribute but are not motivated to be known in the group.

Obviously, any single user can respond to multiple types of motivation but we use the distinctions to inform game design. For example, we'll promote reviewer profiles more actively on the site. We'll be sure to make comparisons of commenting achievements wherever we expose friend rewards, for example, in a user profile. For raters, we'll be sure to make progress towards level improvement prominent. Raters are driven by accomplishment so we want to make progress and accomplishment very visible.

Using this understanding of user roles and motivations, and understanding what our business objectives are, we can make simple games. For example, we can make a game called Sage Reviewer that drives users to create the most reviews in a given week. Winners are featured in a leaderboard and highlighted in special areas on the site. Similarly, we can provide a weekly game that rewards comment creation. Top commenters are featured in a dedicated leaderboard but will not be featured personalities on the site. To ensure reviews and comments are of a high quality, we can provide the Roaring Rater game in which members are rewarded weekly for rating reviews and comments. The cost of participating in the Roaring Rating game is low but the value returned to the site is high. Members easily accomplish useful tasks and gain rewards and status.

Beyond these core games, there are games you provide because the additional cost of doing so is nominal and the return is too great to ignore:

Power Promoter: Become a top promoter by sharing reviews and comments; played weekly and on going. Played weekly and on going, this game is driven by rewards, such as badges.

Model Member: Become a model member by rating reviews, comments, and businesses; played weekly and on going, this game is driven by status represented by levels.

Defining Levels

Levels define status on your site. While many factors inform level design, you can start by considering two factors: site theme, and anticipated usage.

Your theme will provide a rich domain of naming choices. If you run a site for horticulturists, then your first level may be seedling. It's key to remember that levels define status, and unless membership itself is exclusive, then you want members to earn enviable status. Thus, lower levels denote potential, and higher levels represent achievement, respectability. For sites that leverage humor, lower level names may be based on objects of ridicule, for example, Noob. You have to choose wisely though as most people are not incented by insults. For a gadget site, new members can start as newbies, and grow to become gurus. These levels have fundamental and powerful meaning to such an audience. Few members of a gadget community want to be known as a newbie and will probably work to advance.

Unless you've done some analytics, you may not have a good idea of the usage profile of your average user but you can make some guesses. How many times does a member visit a day? How many pages do you they read? Estimate site usage for your ideal average user. Break it down into times members perform the behavior. For example, the average user visits once a day, and reads 5 articles, and makes 1 comment.

Next, you can assign relative point value to each behavior. For example, in Skumo's case, the most important behavior is creating reviews and it's three times as important as comments. It takes longer and can provide many SEO opportunities. Commenting, on the other hand, takes less time than making reviews. That said, readers value comments highly as they can understand by the writing if the reviewer is reasonable and has similar values. Thus, we can state that reviews are worth 50 points, comments 25 points, and ratings can be worth 10 points. Behaviors that require less effort but still provide value are rewarded but are worth less. For example, visits can be 5 points and reading pages can be 5 as well. In this system, it's not enough to show up—you have to play.

Based on these estimates, we can make a simple table:

Behavior	Times	Points	Daily Value
Visit	1	5	5
Read	10	5	50
Review	0	50	0
Comment	1	25	25
Rate	3	10	30
		Daily Total	110

Behavior	Times	Points	Daily Value
		Yearly Total	**5720**

You should remember that in environments like games and web sites, participation is rarely a bell curve. Thus, your level design should recognize that the power law of distribution is probably more relevant in anticipating possible usage. You don't have to solve for this problem, but you should be aware of it.

Level Design Recommendations. Create a profile of a common user and the actions he performs daily and then multiply that by 365 to get a rough idea of accumulated points for a year.

To provide immediate satisfaction and reward, create a level that new users earn when they register.

Make the first few levels easier to attain to incent users to participate more often.

Start with three or four levels and monitor usage. Use analytics to design higher levels.

While Badgeville lets you redefine levels, it is recommended that you avoid restructuring levels so that users are demoted.

Skumo Level Design. Here is an example table that contains starting levels for Skumo based on our anticipated usage calculations, and our understanding of Skumo membership. The theme is hometown because it promotes local businesses. Thus, we use the concept of citizenship as the basis for levels. The first level is Tourist. While it is certainly fun to be a tourist occasionally, it's rarely a compliment. That said, it's not offensive. If you do stay for a while, you can become an Ex-Pat, a long time resident but still not a full citizen. A citizen is someone who belongs but isn't necessarily respected. A Model Citizen belongs and is respected. These starting levels can take Skumo through a year or so until further levels are needed. This approach can be extended by defining more value-laden names or simply extended by adding numeric levels, Model Citizen 1 and so on.

Level Name	Point Range
Model Citizen	12,000–24,999
Citizen	4,000–11,999
Ex-pat	1,000–3,999
Tourist	0–999

Trophies. Trophies visually represent levels, but more importantly, they represent status and reputation in your community. Using your site theme as the basis, make trophies that clearly indicate rank. You can display trophies in user profiles, leader boards, or wherever rank is useful to educate and engage.

Figure 6-4. Skumo trophies

As shown in 0, Skumo's first level is depicted as a camera, which is commonly associated with tourists. Next, Ex-pat is represented by a passport. Citizen is represented by an elderly Greek man, which is an iconic images associated with democracy. Finally, model citizen is represented by a golden crown, worn by citizens with special status.

Defining Rewards, Achievements, and Badges

You reward achievements on your site with virtual goods, like badges, trophies, and points. Rewards can also be tangible, like coupons, discounts, or early access to premium content and features. Points or virtual currency can also be used as rewards.

Skumo's short-term rewards strategy is to rely on low-cost virtual rewards, like attractive badges, and then later support tangible rewards, like discounts or coupons, sponsored by reviewed businesses. We'll share some basic thinking around reward and badge design, and then show you a sample badge collection for Skumo.

Badge Design. When designing badges, you should consider several factors: aesthetics, psychology, and icon design. Visually, badges should be works of art. If badges were clothing, badges are what you wear to look your best. While badges do not cost anything, they aren't free. Users have to earn them. As such, badges represent effort and accomplishment.

Most people enjoy defining or expressing themselves through accomplishment and acquisition. From the clothes they wear to cars they drive, people surround themselves with goods that reflect their personalities. Our virtual selves are no different. The badges we earn are a reflection of ourselves.

Unlike icons, badges can be displayed at various sizes. Your design should be flexible to accommodate this. Also, locked badges can be shown in grayscale with a lock overlaid. You should consider these cases when designing badges.

Reward Collections. A collection is a group of rewards that encourages a single behavior. Most rewards programs use two collection types: ladder and prize. You use the ladder type to reward a user with an ever increasing, more desirable reward for one specific, ongoing behavior, for example, visiting. Ladder rewards leverage personal and friend motivators as they support personal achievement, recognition, and comparison. You use the prize badge type to reward one-time actions like enabling Facebook sharing or participating in a promotion.

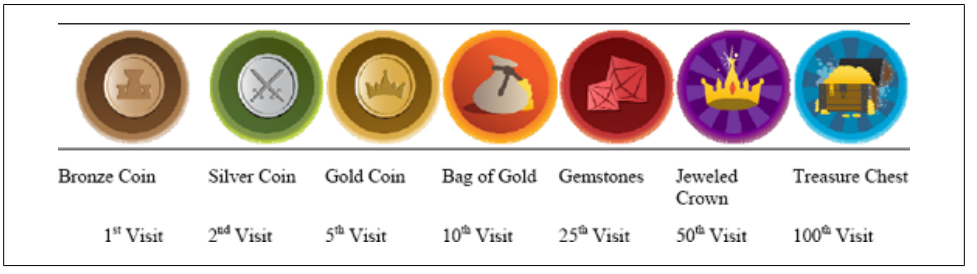


Figure 6-5. An example of a ladder collection for a gaming-themed site that encourages repeat visits

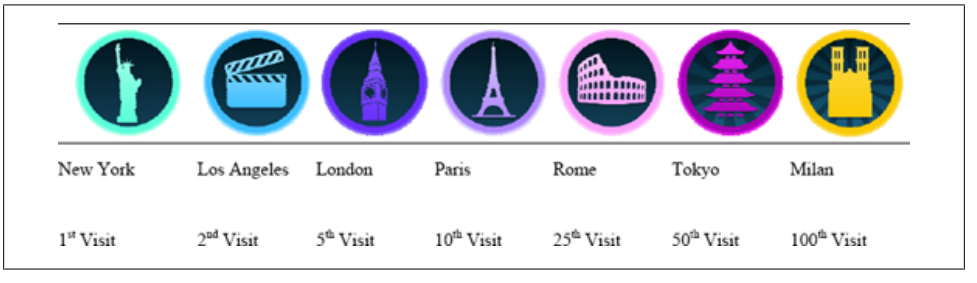


Figure 6-6. A prize collection for a fashion-themed site

Example Rewards Structure. Skumo uses the rewards structure below. You can use a similar table to define your structure.

Behavior	Badge Type	Triggers
Visits	Ladder	7 stages:
		For visits: 1st, 2nd, 5th, 10th, 25th, 50th, 100th
Reading	Ladder	7 stages:
		For reading pages: 10th, 25th, 50th, 100th, 250th, 500th, 1000th
Rating	Ladder	5 stages:
		For rating: 1st, 5th, 10th, 25th, 50th
Commenting	Ladder	5 stages:
		For comments: 1st, 5th, 10th, 25th, 50th

Badge Design Recommendations

- Badges are works of art; invest in good badge design.
- Use the language of your community.
- Use color progression consistently across badge collections, for example, make starter badges bluish, and advanced badges reddish.

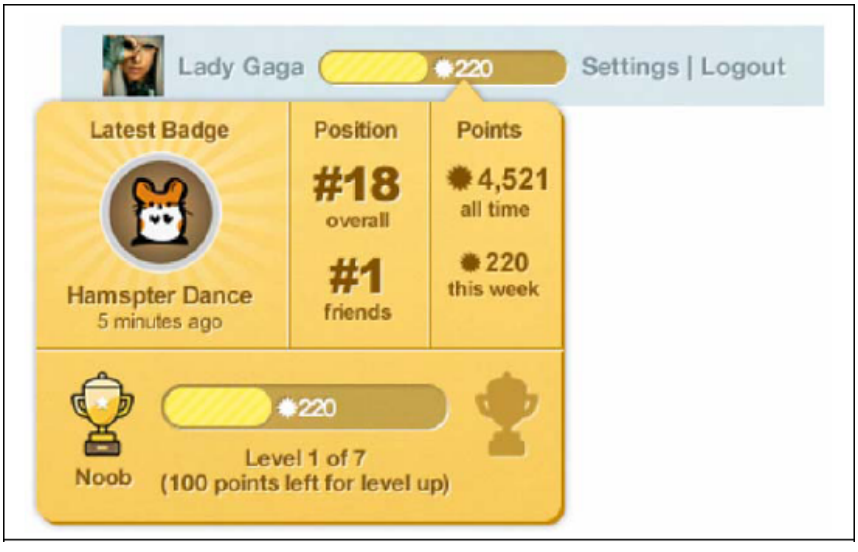


Figure 6-7. Rewards data in a header widget

- Use numeric values on badges when image alone does not convey value.

Rewards UX Design

Your investment in social rewards is wasted if its buried at the bottom of your pages. We recommend displaying the rewards platform as prominently as possible without making it obtrusive. There are many places you can surface rewards data and interactions without losing valuable screen real estate. We'll show examples of the following components and how they can be used:

- Login header
- Leaderboard
- User profile
- Rewards notification
- Inline with content

Login Header. You can use a login header to display to a logged in user, her current level and status. To be space efficient, you can display just information, like a progress bar and points total, to entice the user to explore the control. Hovering over the progress bar displays the user rewards summary. Clicking the user profile image or name opens the user profile page.

Leaderboard. A leaderboard displays your site's top fans for a given time period. A leaderboard is one of the most important tools to educate and encourage your users. Social

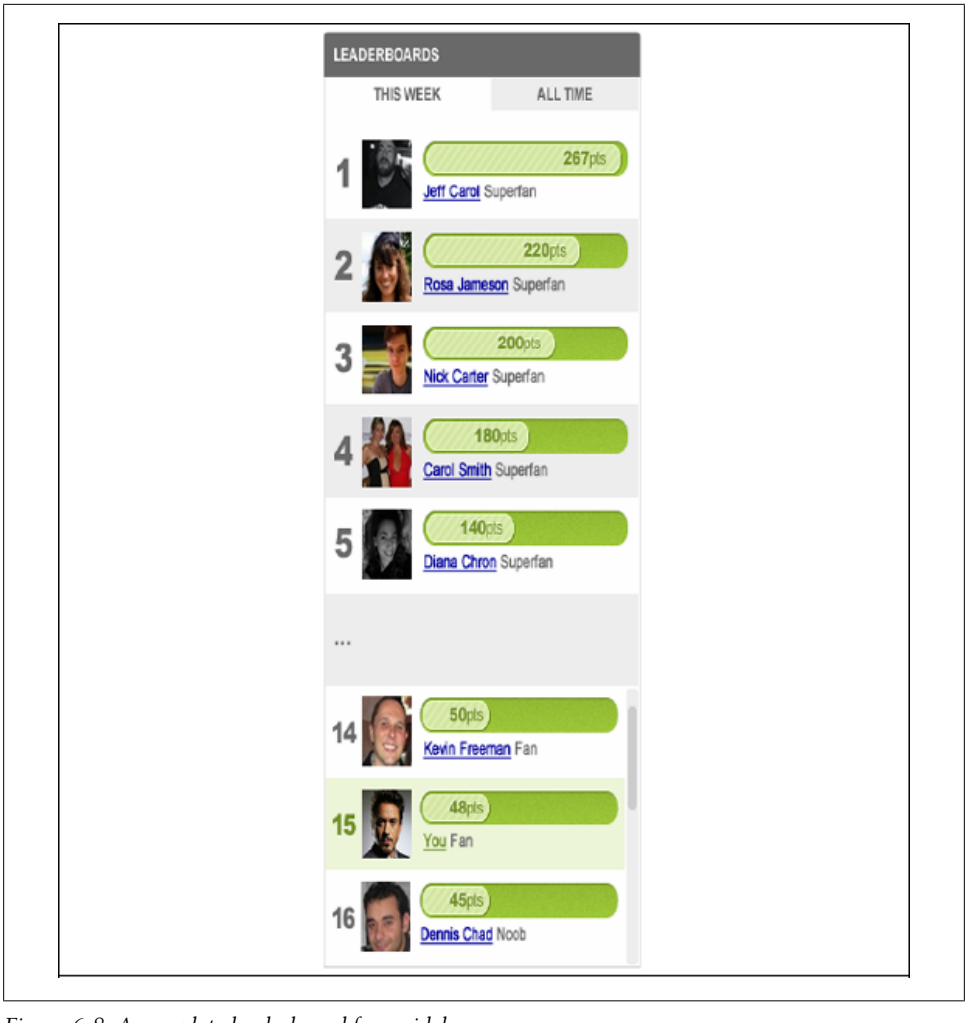


Figure 6-8. A complete leaderboard for a sidebar

and yet competitive, a leaderboard is the public space where members are recognized and rewarded for loyal participation. You can include a leaderboard almost anywhere. You can embed longer leaderboards in sidebars, or shorter ones in content templates.

User Profile. A user profile is a critical component to an awards program. Where a leaderboard is a social scoreboard, a user profile can be a personal scoreboard that features accomplishments and provides guidance about next steps. A user profile should also support achievement sharing. You can design a profile UI that fits into a tab on a leaderboard, or is a complete page, as shown below.

Rewards Notification. Displaying rewards in real time is important. But not annoying your users is critical also. More and more sites are using the bottom of the screen to com-

Figure 6-9. User profile that includes shareable rewards

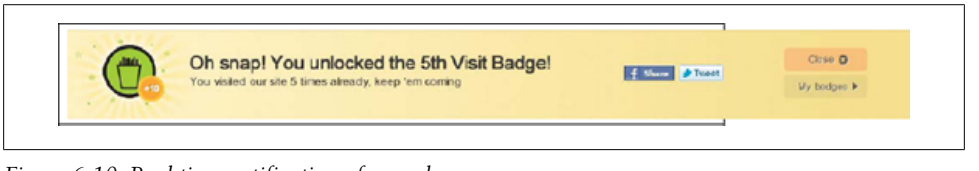


Figure 6-10. Real time notification of awards

municate updates to users. You can use this area to reward users in a subtle but effective way. The rewards notification design can also include social sharing features, and provide access to a user's profile or account settings.

Inline with Content. You can surface rewards content just about anywhere. For example, Philly.com embeds a comment microwidget in their commenting system. User level, trophy, and points appear inside the user information portion of the comment UI.

Leveraging Your Theme

Your site probably has a theme that you can use as the basis for games and rewards. For example, a gaming site may use treasure as the basis for rewards. A first visit may

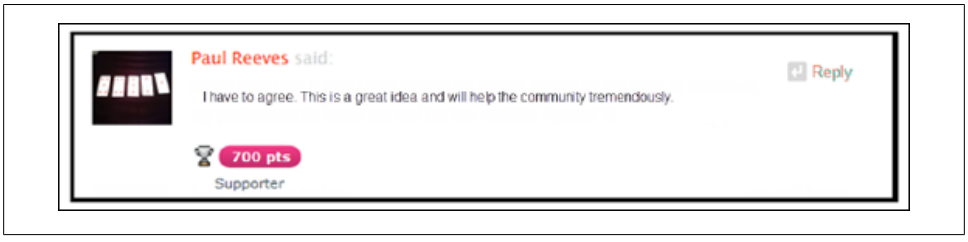


Figure 6-11. Embedding rewards comment inline with other content

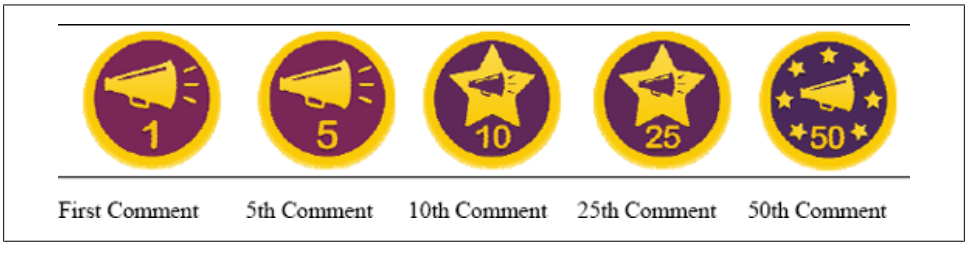


Figure 6-12. Skumo comment badges

receive a bronze coin badge, but the 50th visit may earn a jeweled crown badge. A theme is not required but certainly helps all facets of game and reward design.

Skumo uses the following badge collection to reward commenting. The badge gets more intricate and the explicit value higher:

Developing a Rewards Program

You can implement a Badgeville rewards program by using their plug and play widgets. Or, you can integrate your rewards and loyalty programs deeply into your web site or application using Badgeville's API. This section walks you through the development of a training site, Skumo, which mostly uses the Badgeville API.

Code Examples in this Chapter

Badgeville provides a RESTful API that returns data in JSON format. Thus, you use just about any language to query the API. We provide Ruby samples taken from Skumo.

Step 1: Creating a Rewards Program in the Publisher module

Badgeville provides a Publisher module that allows you to create and monitor all the system objects you need to provide a rewards and loyalty program. You define business objectives, reputation levels, rewards, business rules for awards, and more. You should create the rewards programs you reference in your code, before you run your code.

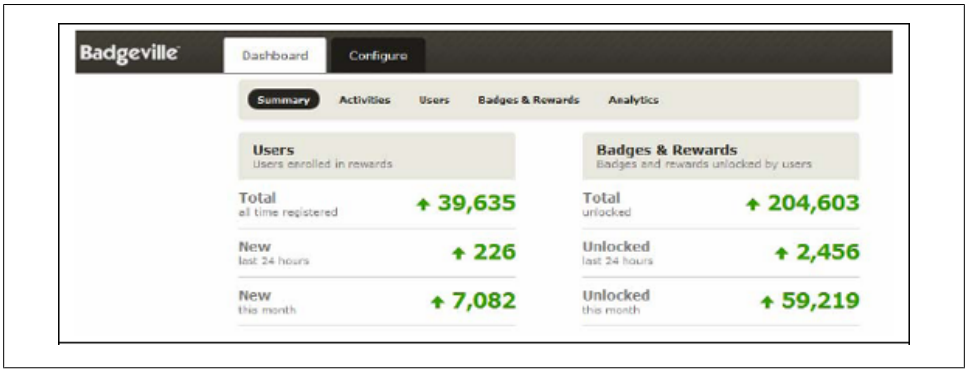


Figure 6-13. Badgeville Publisher module

Before you begin developing, you should create at least one level, a business rule, and a reward. These steps explain how to create a reward.

To create a reward:

1. In the Publisher module, navigate to Configure > Badges and Rewards.
2. Click Create New Reward. The Reward editor appears.
3. Enter a reward name.
4. Enter a description, which appears in the notification when a user earns the reward.
5. Select a behavior, if the reward is associated with a behavior.
6. Select one or more business rules that apply.
7. Click Save.

Once you save a reward, it's available in the system. For brevity, we won't create all required reward objects and we'll move on to discuss project coding.

Step 2: Calling the API

You can make a method to call the API and assign the JSON data to a variable or object. The method accepts a URL argument. Skumo uses the following code to handle calls to the Badgeville API:

```
def bv_send_request(api_url)
 url = URI.parse(URI.escape(api_url))
 req = Net::HTTP::Get.new("#{url.path}?#{url.query}")
 res = Net::HTTP.start(url.host, url.port) { |http| http.request(req) }
 return JSON.parse(res.body)
end
```

We make a generic method that receives a URL, cleans up the URL, makes an HTTP request to the target defined in the URL, and returns data in JSON format. Since we use Ruby, we can either use the standard URI library, or a Ruby gem that handles URLs.

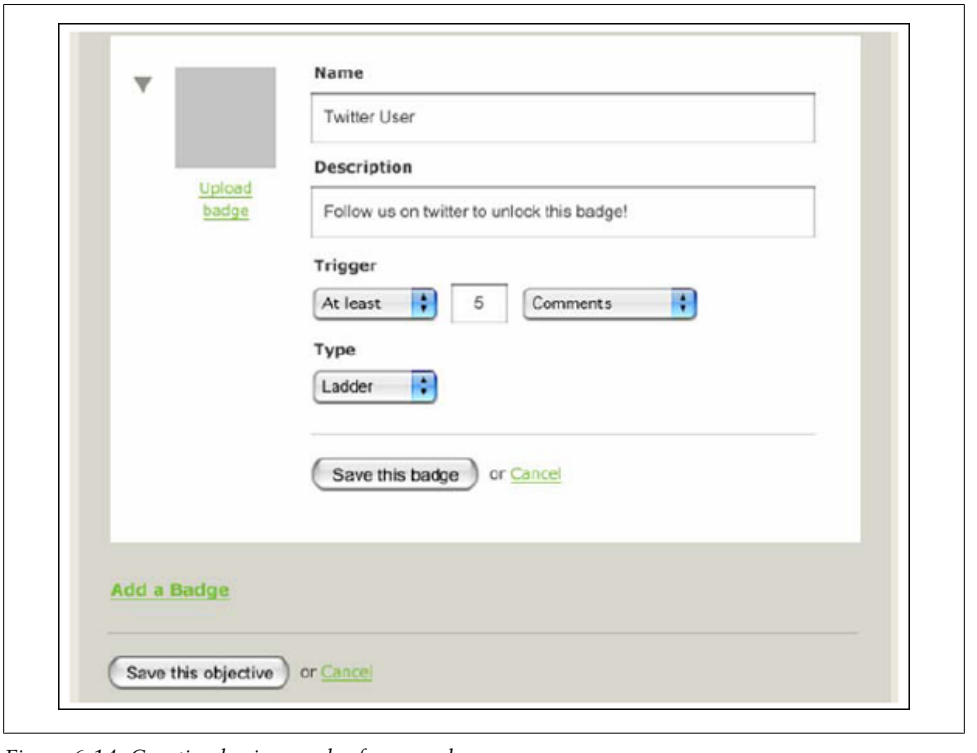


Figure 6-14. Creating business rules for awards

You can easily create your own variation of a generic URL handler that prepares the URL to make a request, sends the request, and receives JSON data. Most modern languages have built-in support for JSON.

Here are some commonly used endpoints you use in URLs:

- Register an activity: *http://badgeville_host/api/publisher_id/users/user_session/user_badges?behavior=behavior*
- Retrieve top fans for a week (to display in a leaderboard): http:// *badgeville_host* api/*publisher_id* /users/week_top_fans
- Retrieve user-specific data to display in a user profile: *http://badgeville_host/api/publisher_id/users/user_session*
- Retrieve the next badge for a specific user: http://*badgeville_host*/api/*publisher_id*/users/*user_session*>/user_badges/next?behavior=*behavior*

Step 3: Parsing JSON Data

The Badgeville API returns data in JSON format, a lightweight data format that is widely adopted due to its ease of use and native browser support. Thus, you need to create a

method to parse data returned from the Badgeville server to extract the information you need. The structure of the JSON data depends on the endpoint you call. For example, this is the call for the leader board:

```
url = http://badgeville_host/api/publisher_id/users/week_top_fans
json = bv_send_request(url)
@bv_users = json["top_users"] if json["result"]
```

In this example, we pass the `week_top_fans` endpoint as a URL to the method we made earlier, `bv_send_request`, and assign the returned JSON data to a variable. When we do the assignment, we confirm that there is a result. If we do have data, we take the `top_users` array in the JSON data and store it in the array `@bv_users`.

Step 4: Register and Track Users

Before you can track user behaviors to reward them, you must be able to identify them. You can use a JavaScript based Tracker that Badgeville provides, or make calls to the API directly. Skumo tracks behaviors of registered users by calling the API, as shown in Step 4: Enabling a Behavior on Your Site.

To track activity for a specific user, you do not have to integrate Badgeville with your user authentication system. You simply create a method that passes existing user information, such as an e-mail and display name, to the Badgeville `update_info` endpoint. Here is a Ruby example that registers a user:

```
def send_information_to_badgeville(user)
  url = "http://badgeville_host/api/publisher_id
/users/update_info?email=#{user.email}&display_name=#{user.name}"
  return bv_send_request(url)
end
```

In this example, we use the method we made earlier, `bv_send_request`, to register and update users. The URL we pass contains user information Badgeville uses for identification and notification purposes. While authentication is handled by your existing system, Badgeville provides additional user validation to ensure the request is legitimate. The `update_info` endpoint returns a Badgeville user session ID you use in subsequent calls to the API to get user-specific data. Here is the code that assigns the value of the session ID to a variable, `bv_session`:

```
...
obj = send_information_to_badgeville(@user)
session["bv_session"] = obj["session"] if session["result"]
...
```

You call the method above and capture the Badgeville session ID for the user from a library that is loaded when a user logs in. This code uses the `send_information_to_badgeville` method to pass the user object to Badgeville and receive a user session ID.

Step 5: Enabling a Behavior on Your Site

Once you know who the current user is, you can enable behavior tracking by either using a Badgeville JavaScript function, or by calling the API to register activity. Skumo tracks review creation, commenting, and rating and other user activities using the API. To do this, you define methods that are called from within your code when necessary, for example, when a user submits a review, or adds a comment to a review.

Here is sample code for capturing pages read:

```
def send_rateComment
 url = " http://badgeville_host/api/publisher_id
/users/{session["bv_session"]}/activities/new?behavior=RateComment&url={request.url}"
 resp = bv_send_request(url)
 @badges = resp["badges"] || []
end
```

In this example, you again pass a URL to the `bv_send_request` method, but the URL is a specific endpoint. In this case, it's the endpoint for rating a comment. You can create one behavior to track all rating, or individual behaviors to track and reward content-specific rating.

Here is sample code for capturing a visit:

```
def send_visit
 url = " http://badgeville_host/api/publisher_id
/users/#{session["bv_session"]}/activities/new?behavior=Visit&url=#{request.url}"))
 resp = bv_send_request(url)
 @badges = resp["badges"] || []
end
...
```

When you register an activity, Badgeville returns a success message in JSON format that includes any badges the user unlocked. Real-time notification of achievements is critical. Instant gratification is a powerful motivator for most users.

Anti-gaming Logic

Badgeville supports anti-gaming in many ways but for generic behaviors, you can define a cool down period for behavior registration. You can define in seconds the length of time that must elapse before a player can get credit for a behavior again.

While most users may not try to game the system, your design must recognize that some players will do so. To protect the integrity of the system, you must design to minimize anti-gaming.

Step 6: Creating a Leaderboard

Skumo displays a leaderboard on its homepage. An out-of-the-box Badgeville leaderboard widget contains three tabs: a site leaderboard, a friends tab with a friend's lead-

erboard, and a user profile. This widget leverages the motivators discussed previously to encourage and motivate all types of members.

Displaying the Leaderboard. After you call the Leaderboard endpoint, you can display Leaderboard data. To do so, you iterate over the users in the JSON object and pull relevant data for each user.

```
<div>
<h2>Top Citizens This Week </h2>
<ul style="list-style-type: none;">
<% @bv_users.each do |user| %>
<li style="float: left; margin-top: 15px; margin-right: 15px;">
<h3 style="font-size: 50px; float: left; margin-right: 5px;">
<%= user["leaderboard_position"] %> </h3>
<img style="width:64px;height:64px;" src=" <%= user["picture"] %> " alt="" />
<p> <%= user["display_name"] %> </p>
<p> <%= user["week_points"] %> week points</p>
</li>
<% end %>
</ul>
</div>
```

The code above illustrates in Ruby a simple version of a leaderboard that displays users by weekly point totals. The Skumo leaderboard below shows a more advanced leaderboard that generates a ranking bar based on total points for the week.

Step 5: Creating a User Profile

After you call the User Profile endpoint, you can display user profile data. You can display a user profile that lists the badges the user unlocked as well as the user's point total and current level:

```
<% if @bv_user %>
    <div style="text-align: center;">
          <h2>Badgeville</h2><br/>
          <img style="width:64px;height:64px;margin:0px 25px" src="<%=
@bv_user["picture"] %>" alt ="" />
<img src="<%= @bv_user["level_picture"] %>" alt ="" />
<span><%= @bv_user["display_name"] %></span><br/>
<span><%= @bv_user["level"] %></span><br/><br/>
<span><%= @bv_user["total_badges"] %> Badges</span><br/>
<span><%= @bv_user["points"] %> Points</span><br/>
<br/>
</div>
<%end%>
```

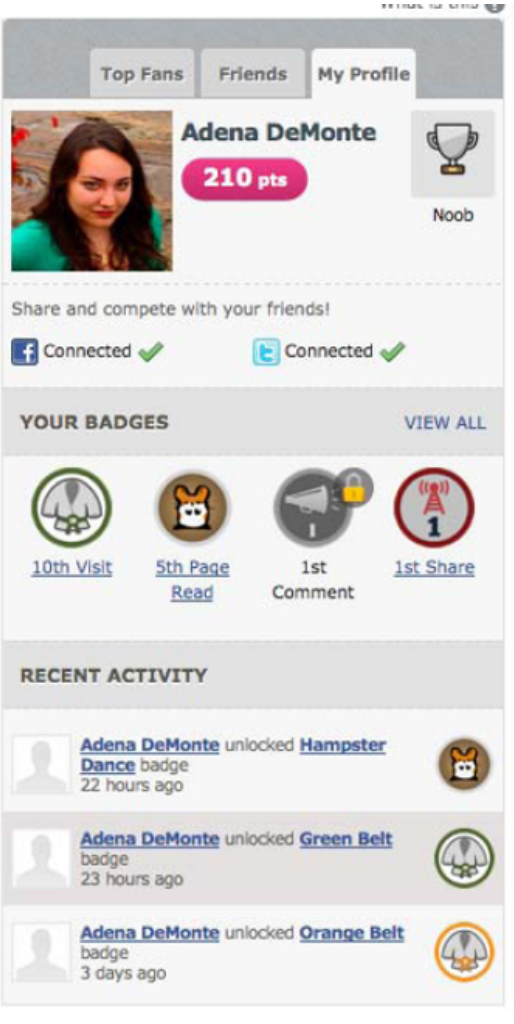
The code above illustrates in Ruby a simple version of a user profile that displays basic user information. The Skumo user profile below shows a more advanced profile that includes a progress bar and a compilation of all badges earned. The badge pop-up displays information about the badge, allows you to share it to Facebook or Twitter, and lets you see who else has earned the badge.

EXAMPLE

TheNextWeb.com

TheNextWeb is an international technology and lifestyle site that focuses on Internet and technology news. With multi-lingual editions and numerous channels such as Apps, Apple, and Viral, this blog based sites serves more than 4.5 million users a month. TheNextWeb uses the Badgeville multi-tab widget with three tabs: Top Fans, Friends, and User Profile.

The Leaderboard widget contains two main sections: user-related information, and site activities. In the user-related portion of the widget, three tabs appear: Top Fans, Friends (Facebook), and Profile. The Top Fans tab displays users by ascending order of status based on Badgeville points earned. The Friends tab shows a user's Facebook friends that are also registered with the site. The Profile tab shows the Badgeville information for a user, for example, badges earned and recent activity.



Example user profile widget from TheNextWeb.com

Step 7: Displaying Rewards

It's important to display rewards in real-time. If with an activity request, the user unlocks a badge, the badge information is returned in the response from Badgeville. You assign badges to a variable to show the notification to the user. Here is an example of notification display. This example assumes you have a DIV for badge display defined in your template.

```
<% if !@badges.blank? %>
 <% @badges.each do |user_badge| -%>
  <% if user_badge["level"].blank? %>
  <div class="badge">
   <img src="<%= user_badge["badge"]["image"] %>"/>
   <p><%=h user_badge["badge"]["name"] %></p>
  </div>
<% else %>
 <div class="badge">
  <img src="<%= user_badge["level"]["trophy_image"] %>"/>
  <p><%=h user_badge["level"]["name"] %></p>
 </div>
 <% end %>
<br class="clear" />
<% end %>
<% end -%>
```

The code example above does not contain all the text from the notification widget, just the code to display the rewards data.

Skumo notifies users when they unlock achievements by displaying a simulated pop-up window in the lower right hand corner of the browser.

Step 8: Creating an Activities Widget

An Activities widget is the pulse of your rewards community. It advertises user achievements to invite and educate your users on what they can do to earn rewards. Badgeville supports two kinds of Activities widgets: site and user.

To display site activity:

```
<script id="badgeville_widget_activities"
src="http://api4.badgeville.com/api/555/widgets/activities"
type="text/javascript"></script>
```

To display user activity:

```
<script id="badgeville_widget_activities" src="http://
api4.badgeville.com/
api/555/widgets/activities?session=currentusersessionid" type="text/
javascript"></script>
```

The user activity example uses the user session info to get activity for a specific user. You can show this information on a user profile page.

Skumo uses a vertical container that updates every few seconds to show an activity. For a large site, you can create a cron job to routinely update a cache that is used to populate the data in a responsive manner.

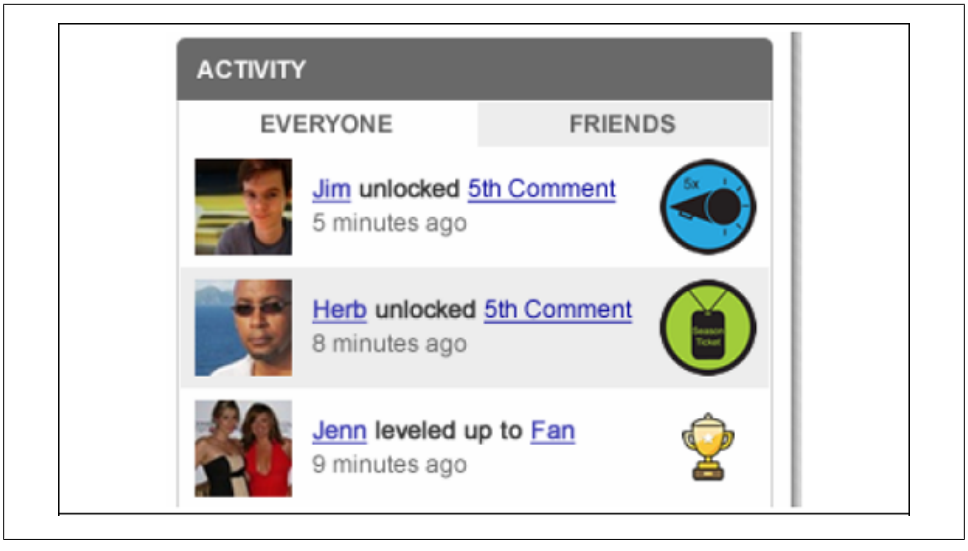


Figure 6-15. Example activity widget

Step 9: Creating a Comment Widget

You can leverage rewards data in other systems on your site. For example, you can include a player's level, trophy, and total points in your comment system using a micro widget. You can create a comment micro widget using the API, or by calling the Badgeville widget endpoint.

To show Badgeville data for a user in comments:

```
<script id="bv_widget_id"
src="http://api4.badgeville.com/api/555/widgets/comments?nickname=Test
User&embed_id=widget_id" type="text/javascript"></script>
```

In this example, you call the comment mirco widget by using the widgets endpoint and specifying the comment widget and the ID of the widget in the page where the comment will appear. The comment widget contains a level image, points total, and level name.

Enabling Social Sharing

A modern rewards program leverages the social web. Here is an example of tracking Facebook sharing by associating the Badgeville `bvCredit` function with the *onClick* event. You can use API calls or JavaScript to track sharing as long as the sharing button is not in an *iFrame*.

```
<a href='example' onclick='bvCredit("Share");'>Facebook Share!</a>
You can give credit for multiple behaviors in one call:
<a href='#' onclick='bvCredit("Share"); bvCredit("FBLike");'>Share!</a>
```

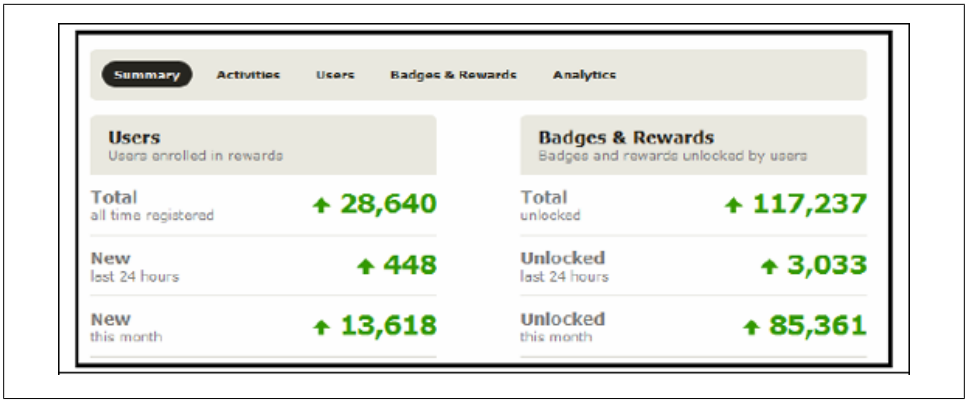


Figure 6-16. Usage statistics

Facebook provides an FBML version of the like button that uses a JavaScript library. The example above uses that approach.

Analytics

Analytics is a key component to a loyalty and rewards program. How else will you know if your programs are working and returning value? The Badgeville Publisher module provides statistics, reporting, and other tools to perform analysis.

Badgeville has an engagement dashboard, answers valuable questions like:

- Are you achieving your site objectives?
- Who are your most valuable users?
- What do your users like?
- What do your users not like?
- Who is sharing the most?
- Who is contributing the most?
- Who is creating the most conversations?

With proper analytics, you can determine which rewards are working and which are not and can adjust accordingly.

Analyzing Sponsored Promotion Success

Here's a simple example. A few weeks after launch, you view a motion chart of behavior trends. You notice that users aren't sharing as much as expected. You decide to encourage sharing by finding a sponsor who pays for the designing and creating of a special badge and advertising for the period of the promotion. Participants sharing reviews during the promotion earn the sponsored badge and a point bonus. After the promo-

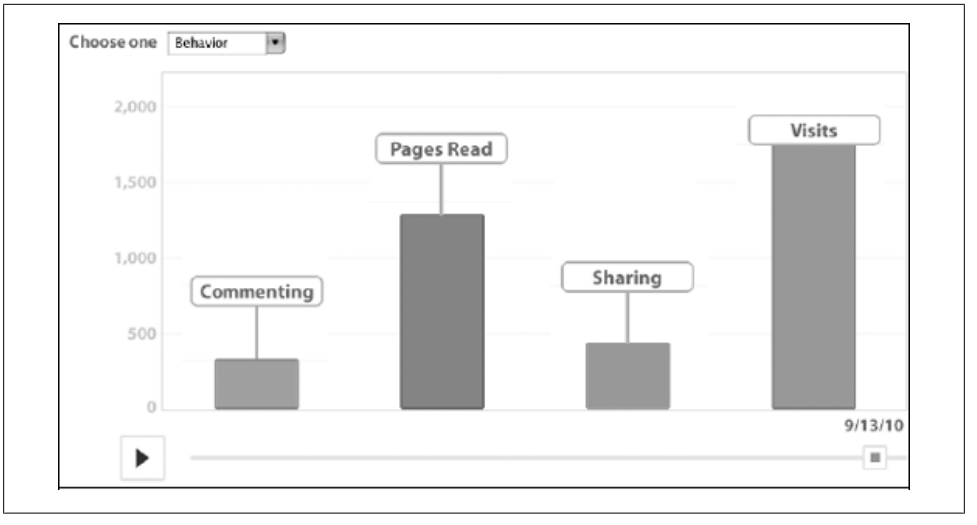


Figure 6-17. Sharing before the promotion

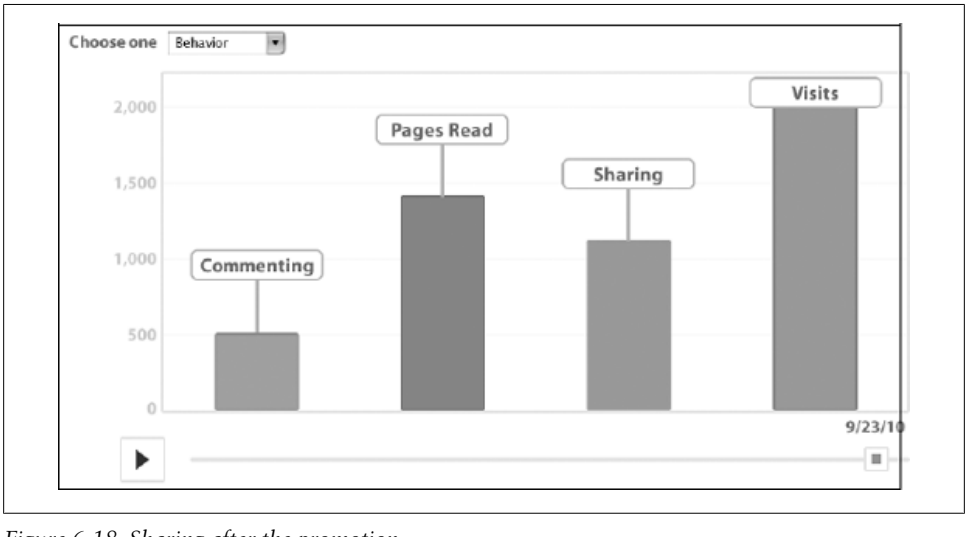


Figure 6-18. Sharing after the promotion

tional period ends, you use a motion chart to track activity trends. You stop the chart just before the promotion starts, see figure below, and you notice the rate of sharing pre-promotion is 23 badges a day.

You watch the chart through the days the promotion was active, and rate of sharing accelerates rapidly (see figure below).

In fact, while other behaviors increased consistently, sharing velocity tripled. To confirm, you run a report and compare two snapshots of sharing totals for two given days

and compare. Notice that you are not comparing page views or guessing user intent. You're analyzing meaningful interactions. Actionable insight is easier to get when the atomic unit of measurement is inherently more meaningful.

Getting Stats from the Badgeville API

If you already have a powerful analytics solution in place, you can query the Badgeville API to get statistics and usage data. Thus, if necessary, you can track the ROI of your loyalty and rewards program outside of Badgeville. From the API documentation, here's a snippet that explains the endpoint to query, and some of the data the service returns to support analytics.

Get statistics

Use this to get statistics related to Badgeville usage on the publisher site.

Route

`/api/publisher_id/publishers/stats`

Receives

`PUBLISHER_ID: The publisher key.`

Returns

A JSON object with these and more:

- Total users registered for that publisher.
- Total number of new user registered in the last 24 hours.
- Total points earned by all the users of that publisher.
- Total points earned by all the users of that publisher, in the last 24 hours.
- Total number of all the shares done by the users of that publisher.
- Total number of shares done by the users of that publisher, in the last 24 hours.
- Total number of badges earned by the users of that publisher.
- Total number of badges earned by the users of that publisher, in the last 24 hours.

```
{"result"=>[boolean], "users"=>[integer], "today_users"=>
[integer], "points"=>[integer], "today_points"=>[integer], "shares"=>[integer],
 "today_shares"=>[integer], "badges"=>[integer], "today_badges"=>[integer]}
```

The Game's Just Beginning

In this short chapter, we tried to show to use game mechanics and a deep understanding of rewards program design to create engaging experiences that return higher yield on customer interactions. Obviously, we barely scratched the surface. But the message

should be clear: users want to be engaged, are flocking to sites that recognize and reward them, and you can provide this engaging experience in a matter of days or weeks. Individuals and companies, who understand gamification and can build it into their platform, will have a significant advantage in the months ahead.

CHAPTER 7

Managing a Virtual Economy with the BigDoor Platform

Managing a virtual economy sounds like an incredibly huge undertaking, especially for developers already taxed with enough. As we've learned from the previous chapters, adding gamification elements to your site can increase loyalty, engagement and monetization. The art and science of balancing sources and sinks is an essential part of implementing a virtual economy, but building out the tools to implement and analyze your virtual economy can be daunting. There are several platforms on the market that can help you quickly set up a virtual economy without having to code all the tools to manage it. In this chapter, we'll take a look at how to use the BigDoor API to quickly set up a virtual economy.

A big advantage to using the BigDoor platform is that it lets you focus your development resources on implementing game mechanics without building out the back-end infrastructure to manage it. There are a number of ways to use the BigDoor platform, including programmatically configuring all the objects in your game system, but in this chapter we'll focus on two aspects: using the admin UI and simple API calls to implement a basic virtual economy and taking a look at some of the widgets available to quickly implement game features such as leaderboards and trophy rooms.

First, we'll take a look at the basics to the BigDoor's API and some examples of how to implement it.

Getting Your Keys to the BigDoor API

The BigDoor API is a self-serve platform; the first step is to set up a free account at bigdoor.com to receive the API keys that you'll need to make calls to their server. The API is REST-based, so any http request to api.bigdoor.com is compatible with BigDoor regardless of the platform or language you are using.

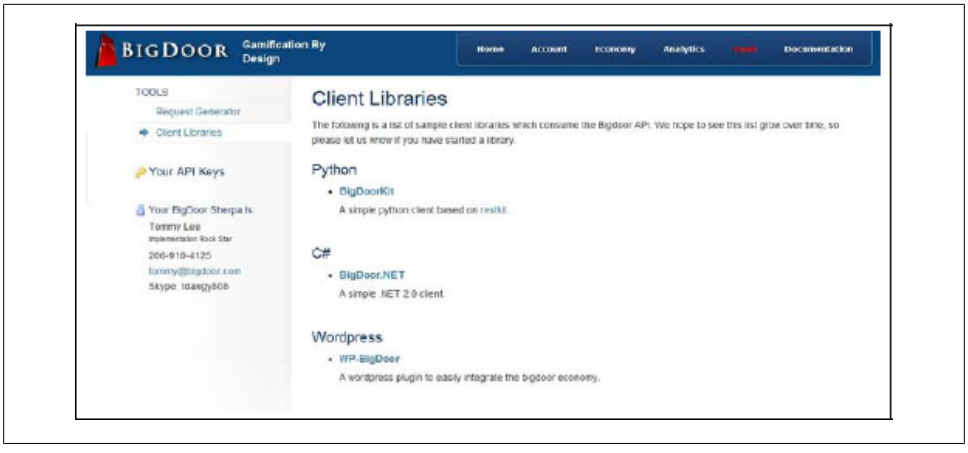


Figure 7-1. BigDoor client libraries

Once you have your API keys you are ready to begin configuring your virtual economy. There are a number of tools available on the site that you can use to implement a virtual economy, including configurable client display widgets and client code libraries.

Client Libraries and the Request Generator

From the admn panel, you can download client libraries to use for API calls, available under the BigDoor Open License. Currently, there are client libraries and SDKs for the following languages:

- python
- php
- ruby
- perl
- iPhone
- android

You can download the client libraries in the Tools section (Figure 7-1).

The Tools section also offers an API Request Generator that allows direct access to the API immediately, so you can test API calls before you start coding (Figure 7-2). Simply select the REST action (in this case a "GET") and the API endpoint you would like to access (we will use "Leader Board"). You will notice that the one pre-filled parameter specifies that data will be returned in JSON, you can pick other formats like XML if you prefer.

You can press the View button to take a look at the call that is being made to the API.

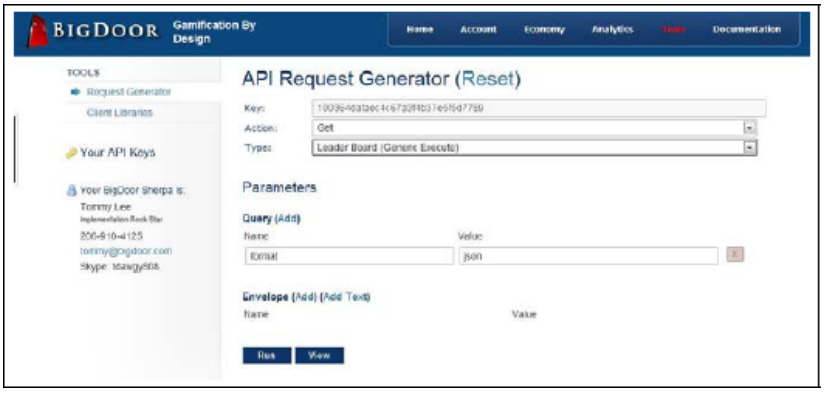


Figure 7-2. API request generator

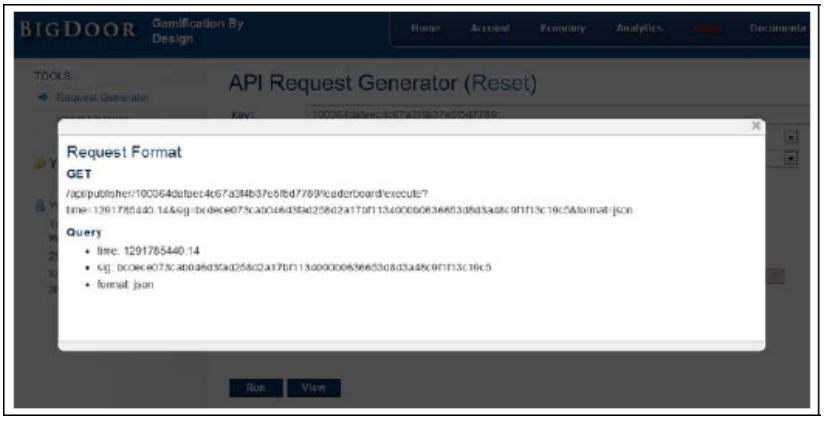


Figure 7-3. Sample API request

Then you can press Run to have the Request Generator make the call to the BigDoor API and return data back. You can view this data in the BigDoor JSON parser that pops up automatically.

As you can see, this action returns general Leader board information based on the data that exists for our virtual economy. We will return to leader boards and how we can pass through a variety of different parameters to return highly targeted data.

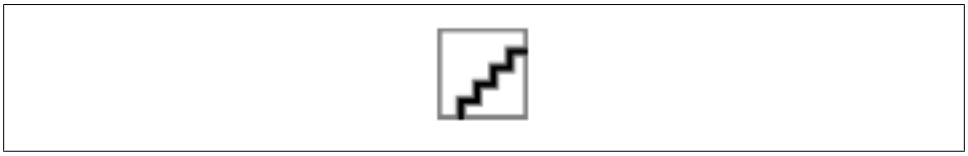


Figure 7-4. Gamification worksheet

Example: Designing a Nonredeemable Currency

The first step to gamifying a site is to think through the basic outline of the game-layer, including the player actions to be tracked. The admin panel offers a Gamification Worksheet (Figure 7-4) to help you map business goals and player actions to the API.

Developing and managing a virtual economy is an ongoing process and not something that can be fully planned out in advance. However, it can be very helpful to have a general roadmap to give you some clarity and goals, even if you end up going in a radically different direction once you start seeing usage patterns and data from your initial implementation. The goal is to create virtuous cycle for players where they can *earn* points and currency for taking actions that are valuable to the online community. Players can then *burn* those points and currency to attain things of real value while they *yearn* for the chance to continue.

To give you an idea of how this might look, let's walk through an example of how we might extend a basic game design for an existing BigDoor implementation, The Cheezburger Network. The Cheezburger Network operates well over 50 (the number is probably higher by the time you are reading this) micro-blog humor websites, including the website failblog.org. Players submit funny news articles and pictures about big failures and then vote for their favorite content. The site current implements badges and trophy cases for users who upload well liked content, but if they wanted to drive deeper engagement with the site and the content, we could use a virtual currency to implement a leveling system. Let's walkthrough how this could be done with LOL posts.

We could assign a nonredeemable voting currency that allows us to track each time a visitor the site votes on one of the LOL posts. We would first decide on an initial point structure voting actions, as seen in the example worksheet in Figure 7-6.

We could then define points earned for voting to design a series of levels, so that players would earn a Trophy each time earned enough points to attain a new level.

That yields a basic game layer design that would let visitors to Failblog can earn trophies for voting on LOL posts.

Next we'll discuss how we put this plan into action using the BigDoor platform. First we'll look at how to set up new players by looking at BigDoor's User implementation.

Figure 7-5. Voting on Failblog

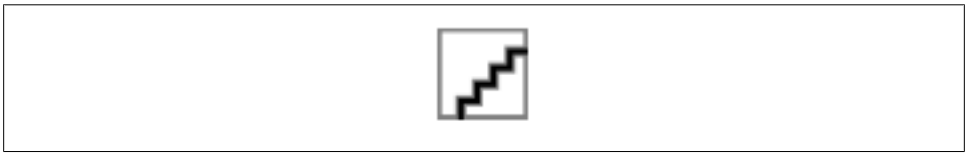


Figure 7-6. Voting action worksheet

Figure 7-7. Voting level ladder

Implement now for future optimization and iteration

Because building and managing a virtual economy is an on-going process, you want to be sure that whatever system you implement will allow you to make the constant tweaking, additions and subtractions from the player experience that are required for a compelling game-layer. These decisions should all be data-driven and based on your community.

One advantage to using a gamification platform to implement game mechanics is that you can quickly optimize and iterate on your virtual economy, without having to recode the game layer. This will be easier if you follow some simple implementation best practices.

As you build in API calls to the system, and tie them to player actions, you want to think about how you can be generic in your approach and allow as much of the game

logic to reside in the platform as possible. For example, rather than hard code rewards or points, you should code for transactions that are processed in the background by BigDoor using the game logic you have provided in the configuration tools.

Registering Users

BigDoor abstracts all of a player's game statistics in the API, so that you don't have to pass or store any personally identifiable information in the API and backend databases. You can maintain all of your player registration on your own system by creating a table that associates your player ids with a BigDoor Global User ID (GUID). You can then pass the GUID back and forth to the BigDoor API and match responses with your User ID table, separating your player's personal information on your server from their BigDoor user game information.

However, if you would like to pass additional information about your users to the BigDoor platform (such as location, social graph, etc.) you can store that information on the User record. This can useful as it later allows to provide highly contextual game elements to your players.

Here's an example of an API call that creates a new BigDoor User ID "Patrick" with the additional information that he lives in Seattle.

```
[Insert example API call]
```

Users are created on the fly in the BigDoor API. If you don't create a User ID before a Transaction Group is executed, that Transaction Group will automatically be associated with a GUID, a nice feature if you want to tokenize your user registration.

Currencies

Let's start with Define Currencies.

Currencies are objects in the BigDoor API that can be credit and debited for any user based on logic supplied by you and managed by BigDoor. These can range from a basic experience point system that runs behind the scenes to full-blown, dollar-backed virtual currency that can be transacted for goods by your users. The types of currencies you configure and use for your virtual economy will depend entirely on your objectives and the type of game-layer you are looking to create.

You can create *n* number of currencies and there are several different currency types to meet a variety of use-cases.

Types of Currencies

These currency types can be divided into redeemable and non-redeemable currencies and then further sub-divided into three primary currency types: purchase currencies,

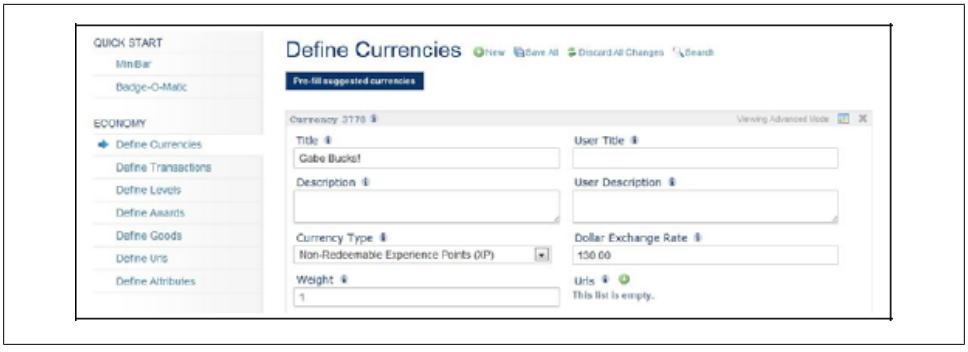


Figure 7-8. Defining a new currency

Figure 7-9. Currency options

> reward points, and hybrid currencies. Redeemable currency can be redeemed or transacted by end-users for Goods (which we discuss a little later in the chapter) whereas non-redeemable currency can never be exchanged or transacted. Purchase currency is 100% dollar-backed virtual currency that can only be earned by players for taking actions that accrues actual dollars back to you, the publisher (such as direct payments, completing CPA offers, or watching video advertisements). Reward currency is a point system that has no actual dollar value associated with it and can awarded to end-users for any action taken. And, hybrid currency allows you to reward any action taken with a virtual currency that may have some actual dollar value.

So for our Failblog example, we are going to create a simple non-redeemable tracking point currency with the Title "Voting Points." We'll add brief description in the Description field to remind us later what this point system is being used for. We can ignore the User Title and User Description fields since we don't plan on displaying this currency to end-users in our system—we'll only use it track voting level-ups.

For future reference, make a note of the Currency ID circled above.

Now that we have created a simple points system that can track each time a user to failblog.org votes on an LOL post. We'll use these points to level-up users and award them trophies as they progress through a pre-defined trophy track.

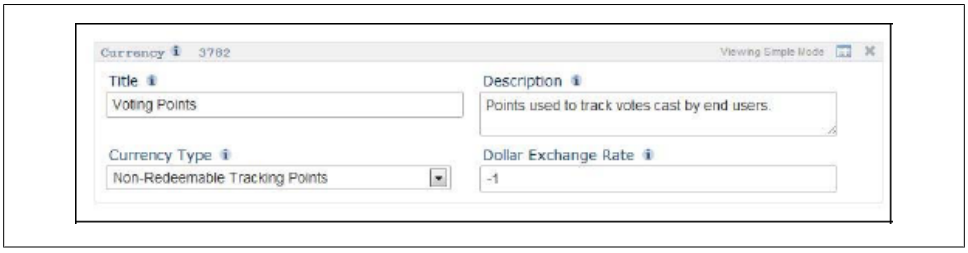


Figure 7-10. The new Non-Redeemable Tracking Points currency

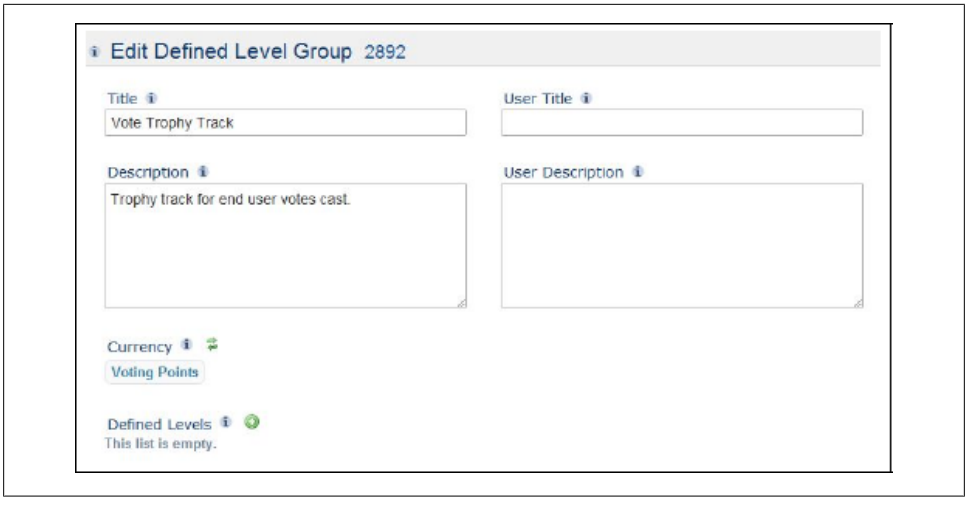


Figure 7-11. Defining new levels

Levels and reward tracks

Moving on to the Define Levels tool in the Economy section, we can add a new Level Group for Failblog votes. First we'll enter "Vote Trophy Track" in the Title field and add a Description for our internal reference. Then we will to tie-in our newly created Voting Points currency.

Next we will create our first level in this Level Group. We'll call it "Voting Level 1" in the Title field and add an internal Description. Then we are going to enter "I Have An Opinion" in the User Title field and "Earned by Voting 10 Times" in the User Description field. Because we want players to earn this Trophy on their 10th vote (and because Voting Points will be awarded on a 1:1 ratio with actual votes), we will enter the value "10" into the Threshold field.

Finally we will add an image URL to this first level so our players can have a cool visual representation of their achievement. We can pick from the existing URLs already in the system, or enter a new URL. Let's add a new URL. We'll Title it

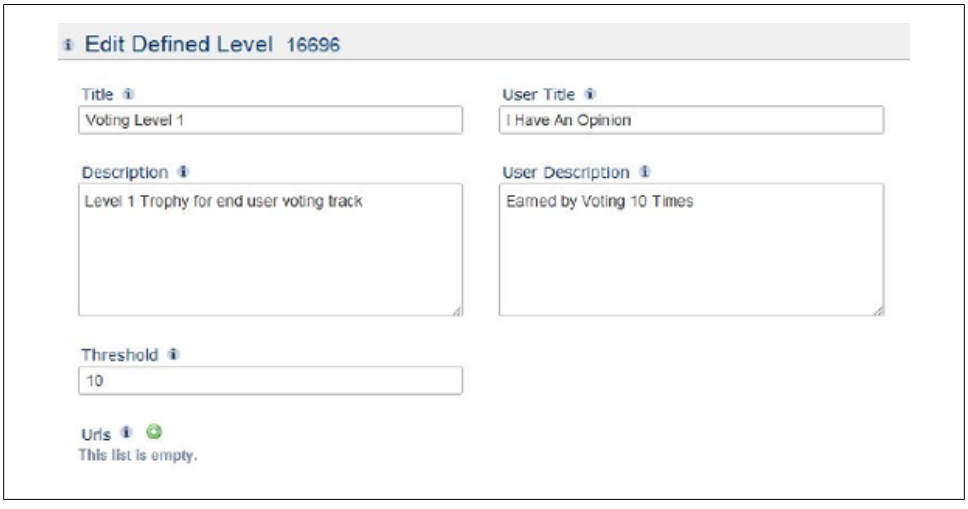


Figure 7-12. A newly defined voting level

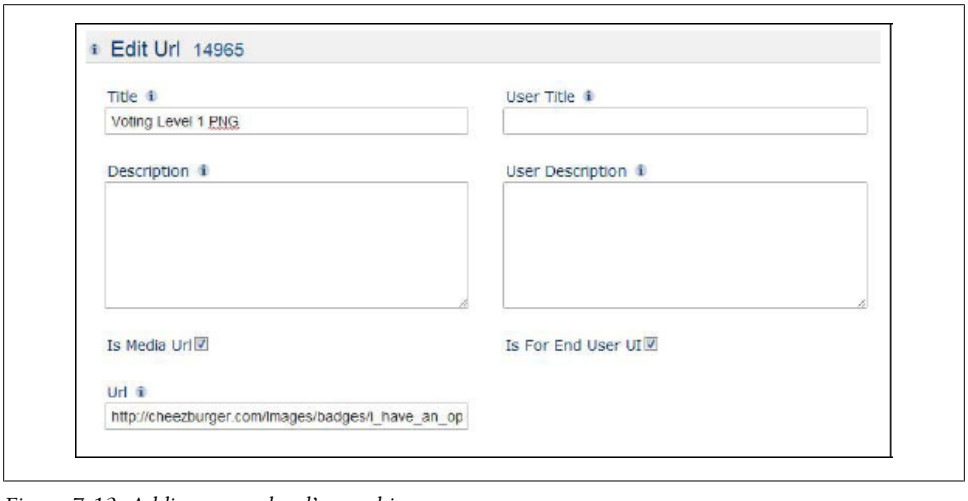


Figure 7-13. Adding a new level's graphic

`Voting Level 1 PNG`, check the box "Is For End User UI" and enter the following into the URL field, *http://cheezburger.com/Images/badges/i_have_an_opinion_medium.png*.

Now our URL is available to choose from in the list.

We've now associated the image URL to Level 1. We should note the circled Level ID. Keep in mind that we note the circled ID so we can use it form the query to call the API and get analytics on that particular level.

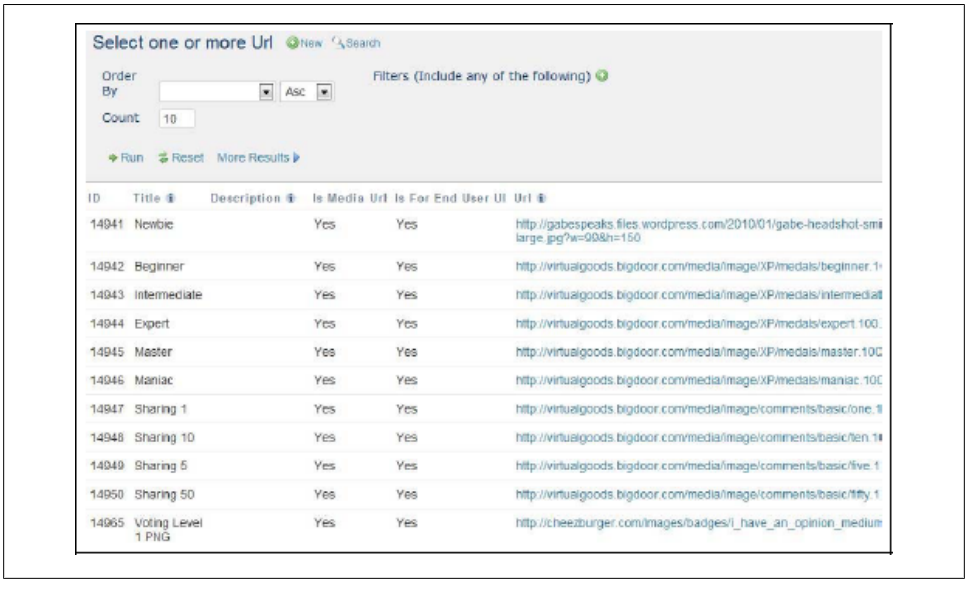


Figure 7-14. Configuring the new level's image

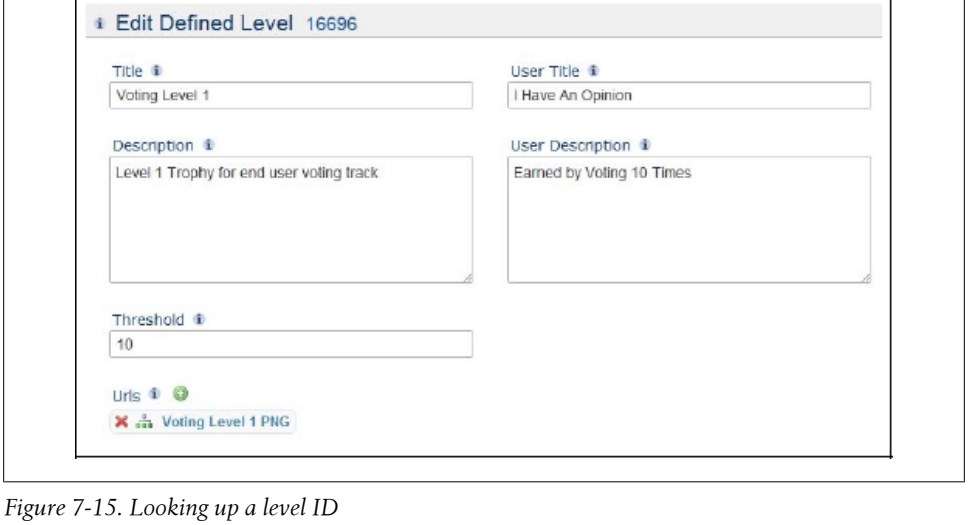


Figure 7-15. Looking up a level ID

Add now we can add a couple more levels to our Trophy Track. Again, let's make a note of the circled Level Group ID.

Now that we have Voting Points and Trophy track in place, let's tie this into the Failblog site by mapping voting actions to the game logic stored in the API.

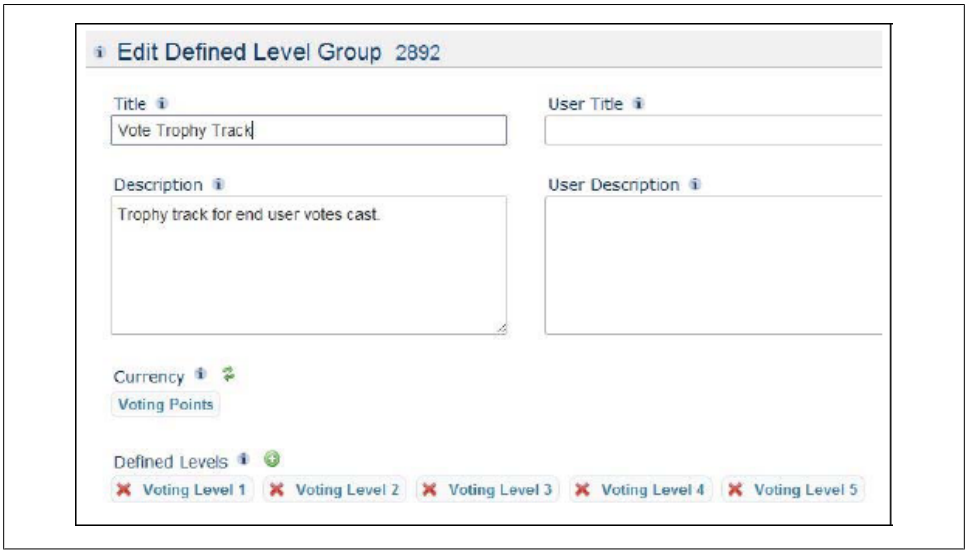


Figure 7-16. Configuring the level group

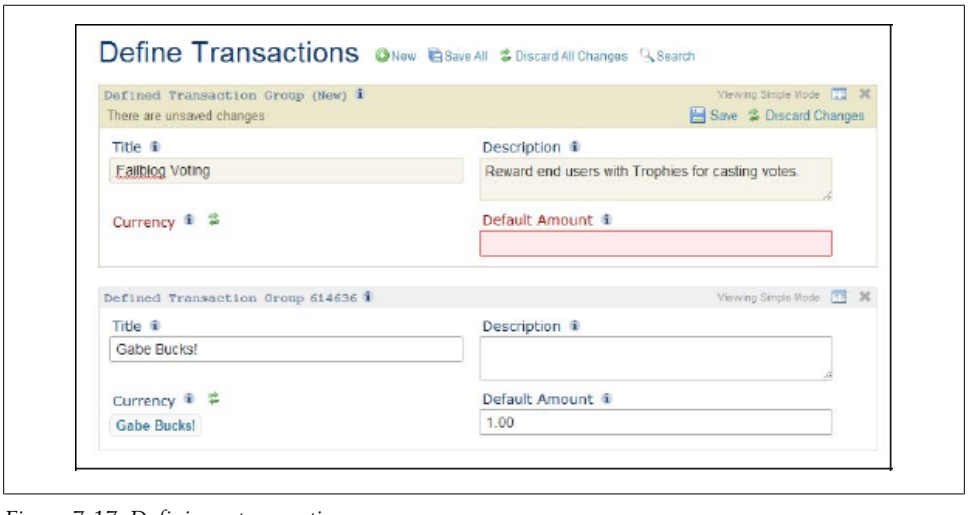


Figure 7-17. Defining a transaction

Transaction Groups

The Define Transactions tool in the Economy section is where the rubber meets the road. This is where we flesh out the game-logic and tie player actions to points and rewards. For our example, we will create a new Transaction Group with Title "Failblog Voting".

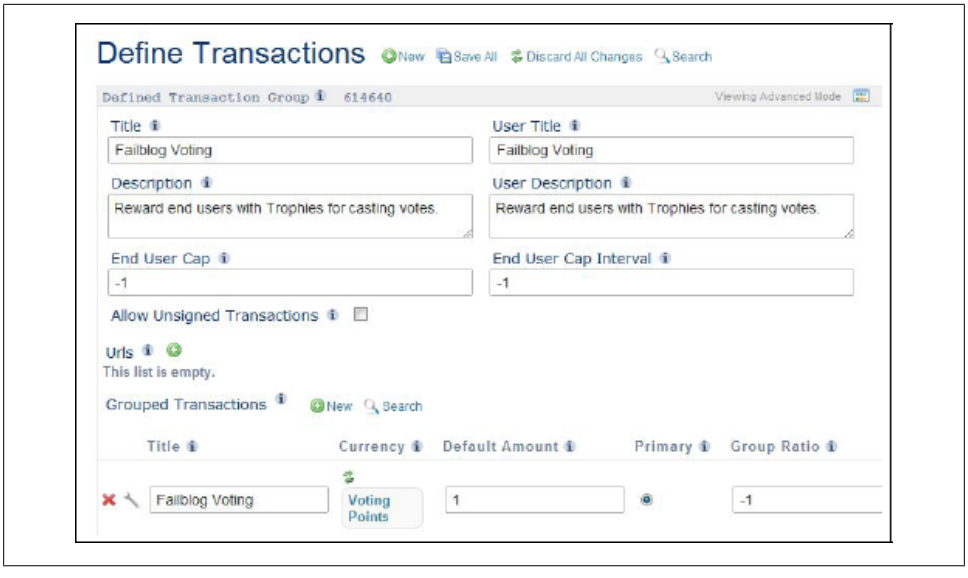


Figure 7-18. Configuring the transaction group

In the Advanced View we can click "New" next to Grouped Transactions to add a new nested Transaction our newly created Transaction Group. We will give our new Transaction the Title "Voting Points" and associate our Voting Points currency to this transaction. Because players earn currency in this Transaction we'll check the box for "Is Source". In Table 7-2 we decided to give players 1 Voting Point for each vote, so we'll define that on this screen by entering the value "1" into the field "Default Amount".

Now we have a specific Transaction listed in our Transaction Group that awards one Voting Point every time our API is told to execute Transaction Group ID ##### (circled above).

Now when a player takes the action voting we can make the following API call to make sure the player gets their points and rewards.

```
[insert example API call]
```

You'll see when the player takes the first voting action the API returns the following response body:

```
(INSERT RESPONSE BODY FOR 1).
```

Here is when the player has taken 10 voting actions and he are rewarded with his first trophy:

```
(Note the circled URL object) (INSERT RESPONSE BODY FOR 10)
```

As soon as a player accrues ten Voting Points the API will grant that player Voting Level 1. The API will then pass the Voting Level 1 PNG URL back to your player-facing UI, and your player will see his cool new trophy.

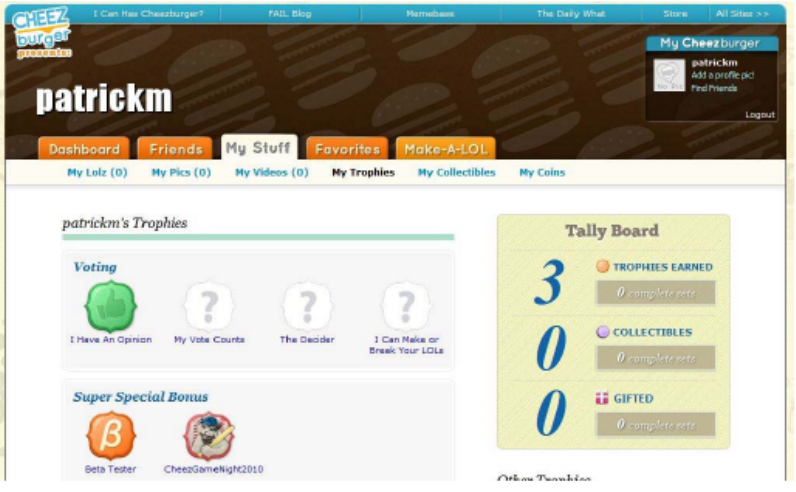


Figure 7-19. The resulting trophy room

Unlocking Content After a Level-Up

Although we are using an image URL for this example, the BigDoor API is flexible enough to accept any type of URL object, or none at all, for level progressions In addition to trophies and badges, leveling can be used for a variety of purposes that includes unlocking premium content or VIP areas, assigning power-ups or features and of course rewarding player achievement with coupon codes or other tangible goods.

Tune your virtual economy

By spending a little time configuring your virtual economy with all of your game logic you can now easily add new game elements without having to touch the codebase or modify the above API call. While Failblog isn't employing these exact techniques, the example is exercise for thinking through how to solve two hypothetical business objectives: overall engagement on the site and monetization through virtual currency and micro-transactions.

To expand on this example, we could add a second currency to the system: a new meta-Experience Point (XP) system and Trophy track to increase overall engagement with a variety of player actions across the site. Additionally we can a second virtual currency to monetize those engaged players. This virtual currency can either be purchased, earned or rewarded depending on the publishers wants and needs. Next, we'll walk through how we could add these features to tune the Failblog Voting Transaction Group we created above.

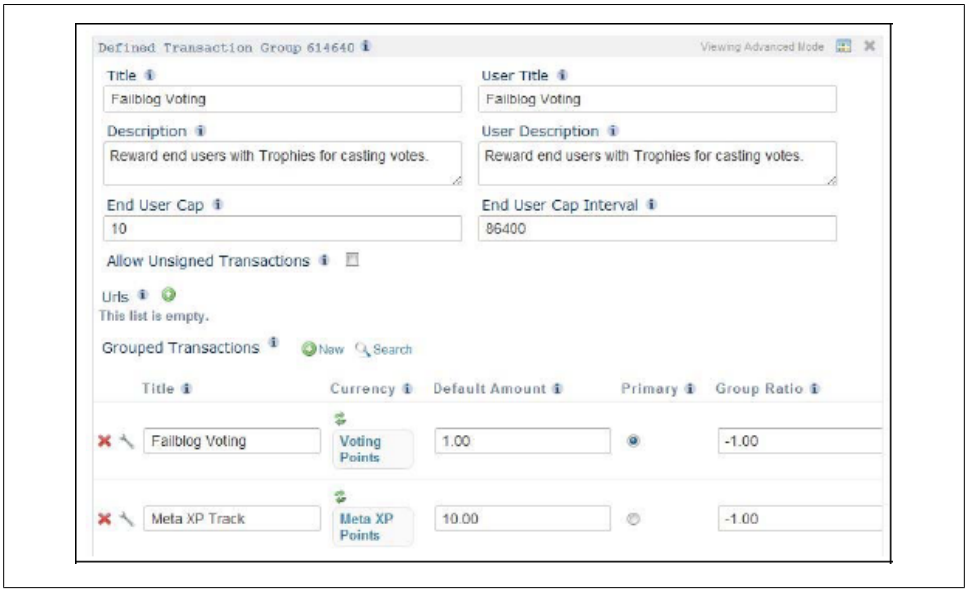


Figure 7-20. Adding an experience point system

Now that we have this virtual currency we have to figure out how we can monetize it. Three classic ways of creating sources of virtual currency are through incentivizing participation and engagement with advertising on your site (e.g. video pre-roll). The second way to create a source of virtual currency are by rewarding actions that would otherwise have cost you more money—like player acquisition and retention. For example if you spend $3 to acquire a new player through traditional channels you can offer players directly a $1.50 in virtual currency for signing up for an account and cut your player acquisition costs in half. Finally, by selling currency to your players through direct payment and alt-pay methods (e.g. CPA offers, surveys, etc.).

Having figured out how players can earn virtual currency, now we need to use BigDoor to help our players burn it. There are a lot of legal issues involved in using redeemable currencies; BigDoor has developed a legally compliant process that allows you to avoid most U.S. regulatory hurdles.

When you use BigDoor you can quickly and easily develop this earn, burn cycle to encourage players to yearn for more virtual currency thereby increasing your monetization opportunities and player engagement and happiness all at the same time, creating a truly virtuous cycle.

Since we've tied points to player actions on the site, we can use analytics to spot areas where we need optimize the virtual economy and game layer—this is called tuning the economy. When tuning an economy managed by BigDoor, you can daisy-chain *n* number of nested transactions to a transaction group, and modify the amount of points awarded for each currency in the transaction group on the fly. You can constantly add

Figure 7-21. Tuning the experience points

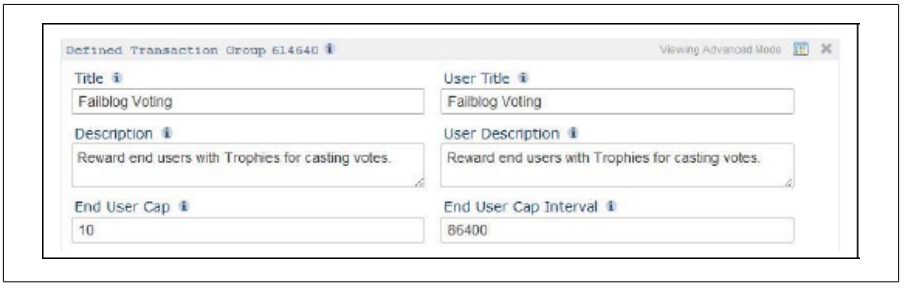


Figure 7-22. Adding a frequency cap

or delete portions of the game layer that aren't working based on the data you've received from the API and you can endlessly adjust the point distributions as needed to keep your virtual economy in balance.

Using Frequency Caps to Limit "Gaming" of Your Economy

To help keep people from "gaming" your virtual economy, you can also set frequency caps to limit how often this transaction group can be executed for a given player. In this case, we want to encourage voting but we don't want people to advance through our whole trophy track in one sitting. So we are going to start with a frequency cap of 10 times this Transaction Group can be executed per day. This will allow players to earn just enough voting points to attain their first trophy, but no more. To do this we will enter a value of "10" into the End User Cap Field and a value of "86400" (the number of seconds in a day, yes that's how precise the BigDoor tools can be) to the End User Cap Interval fields.

Of course, we can always change either value at any time if the data suggests we are being too permissive or to the contrary too preventative. Or we can turn frequency capping off entirely by entering a value of –1 to both the End User Cap and End User Cap Interval fields.

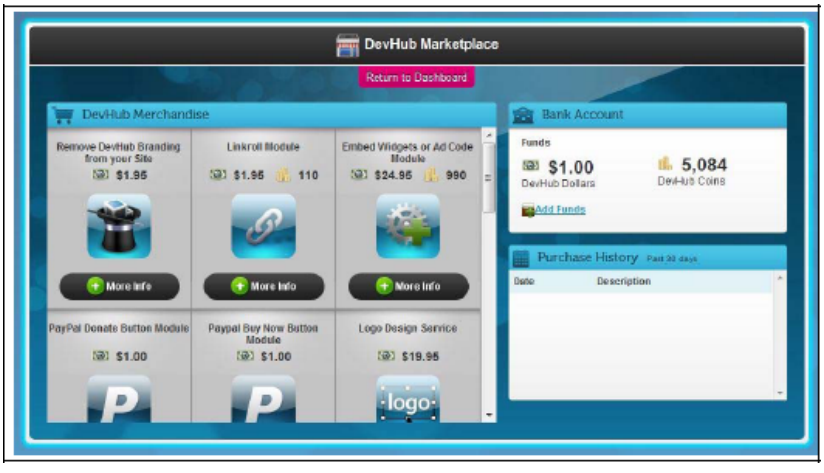


Figure 7-23. The DevHub marketplace

Examples Of A Robust Virtual Economy

The preceding section covered the very basics of implementing a virtual economy using BigDoor. There are many more complex ways to build on these mechanics to drive deeper engagement. Next we'll examine the structure of a few websites with much more complicated virtual economies.

Goods and Monetization

DevHub is a BigDoor-powered site that has implemented a robust virtual economy, DevHub.com. DevHub gamified their website builder and web hosting product. The site utilizes multiple currencies that are visible to players (and many more that are not visible to players) including XP, DevHub Coins and DevHub Dollars. XP and DevHub Coins are earned by players for taken actions that are valuable to them and the community, usually building out or adding to their DevHub hosted website. DevHub Dollars are purchased through direct payments or earned by completing alt-pay offers and surveys.

As you've learned, managing sources and sinks is essential to a healthy virtual economy. One way DevHub balances their virtual economy is through their virtual good marketplace. The marketplace allows players to *burn* both DevHub Coins and DevHub Dollars.

There are a couple things that are immediately noteworthy about the DevHub marketplace. The first is that some of the *virtual* goods available are real tangible goods (like t-shirts) and services (logo design). Because the BigDoor API can be used to create

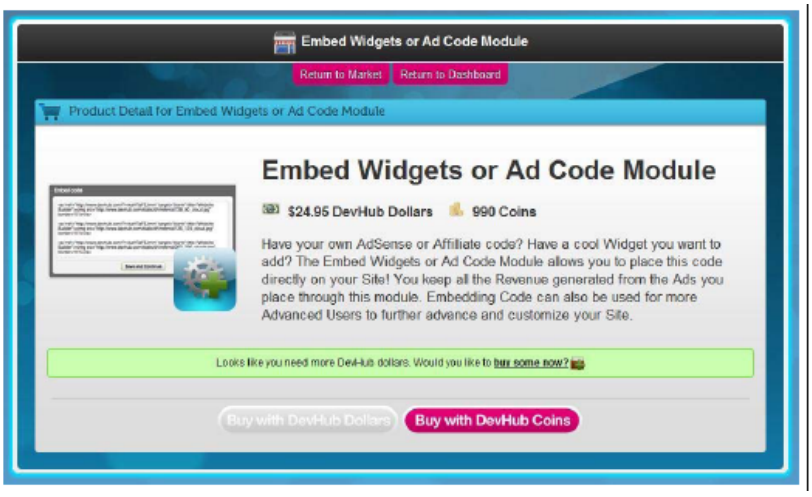


Figure 7-24. Selling premium features with a virtual currency

a Good out of any URL object, players can spend their currency on the things they *yearn* for: real goods and services, virtual goods and unlockable special features.

The second thing you'll notice is that DevHub displays prices in both DevHub Coins and DevHub Dollars. Both of these currencies are redeemable even though DevHub Dollars is the only dollar-backed currency. Having two redeemable currencies gives DevHub's players a choice: they can spend additional time on site to earn a sufficient number of DevHub Coins to purchase the good they desire, or they can buy DevHub Dollars and get that good right away. This type of system fosters both engagement with the site and a sense of autonomy within DevHub's players, a key intrinsic motivator.

As a testament to the power of gamification to affect the bottom line, three months after implementing their virtual economy DevHub saw virtual good revenue comprise 30% of overall company revenues.