# Chapter Three
# What is it Good For?

So, for the first time in the history of the videogame form, people who aren't programmers or corporations can easily make and distribute games. But why would they want to? Why make a game—especially when there already exist the means to write stories, play songs, film yourself for YouTube? What can we do with games that we can't do with those forms?

To begin, let's define what a game is.

You've played games and you have assumptions about what they are. Maybe when you read *game* you imagine a videogame; maybe when you imagine a videogame you imagine a big-budget run-jump-shoot game. Maybe you imagine Tetris. Since I'm more interested in games, digital and otherwise, that don't resemble games that already exist, I think a fresh definition is in order. I also think it's worthwhile to have a definition that isn't specific to digital games, because I'm interested in the commonalities between digital and non-digital games, and in connecting videogames to that much older tradition.

So here's my definition:

> A game is an experience created by rules.

That's pretty broad, huh? I'm interested in as inclusive a definition as possible, though you might argue that mine is too broad: for example, you can use it to describe getting

stuck in a traffic jam or paying your taxes. A tax form is nothing but a series of rules you follow to produce a final number, after all. But is it useful to think about your taxes as a game?[33] Not really. Do the rules on a tax form really create a strong experience, or are they just a method for producing a number?

A game is an experience, and that experience has a certain character. Maybe a game is a story, or maybe it's the experience of control giving way to panic giving way to relief. Maybe it's about taking something and making it grow bigger and bigger and bigger, or maybe it's about two rivals, equally matched, each trying to out-guess the other's plans. The experience that we identify as a game has character, and we can talk about what that experience is.

And if we're discussing an experience, then that implies someone is there to have that experience, someone we refer to as a *player*. We can't talk about a game without talking about the experience of the player playing that game, even if the playing experience we're talking about is often our own.

The experience we call a game is created by the interaction between different rules, but the rules themselves aren't the game, the interaction is! A game can't exist without a player or players: someone needs to be engaging with the rules for the experience to happen.

How does that work? Consider a game of Tag. Rules: One player is IT, and must tag as many of the other players as possible with a touch. Each of those other players is SAFE when she touches this gnarled-up oak tree. You can see the way the interaction between those two rules creates an interesting (and volatile) dynamic. The players who aren't IT want to reach the tree, but the player who is IT wants to stop them.

You can imagine a situation where the IT player is standing between two other players—one to her left, one to her right—and the SAFEty of the tree. Maybe one of them will make a break for the tree, maybe IT will be forced to pick one of the two to chase while the other gets to make a run at the tree, maybe a fourth player will take advantage of IT's distraction to make a run at the tree from behind. When we talk about a game of Tag, we're talking about this experience. But this situation (and it's a good, tense one) isn't explicitly defined anywhere in the rules. However, notice how these rules guide the creation of that situation. The rules set the players in opposition to each other, give most of the players a goal, and give the other player a reason to intervene, creating a tense dynamic.

What if we were to take either of these rules away: the SAFE location or the player who's IT? Without a SAFE location, players have no reason to stay nearby and interact with the other players, especially the IT player. The ideal strategy to avoid IT would be to go as far away as possible, and that breaks the tension and hence the experience of the game. What if there was no IT player? Then it'd just be people running around, and while a bunch of people running around has value, it doesn't have the character or dynamic of a game.

But there's certainly room to change the details of the rules. Tag, being a folk game, has been played by many people in many places with many, many different versions of the rules. In one version, a player might be done once she's tagged the SAFE tree. As more and more players tag the tree and leave the game, the players who are less fast become greater and greater targets because the IT player can focus less on monitoring the tree and more on pursuing them.

Alternately, what if a player who touches the tree isn't permanently safe—what if players are only allowed to be in contact with the tree for five minutes at a time? That keeps

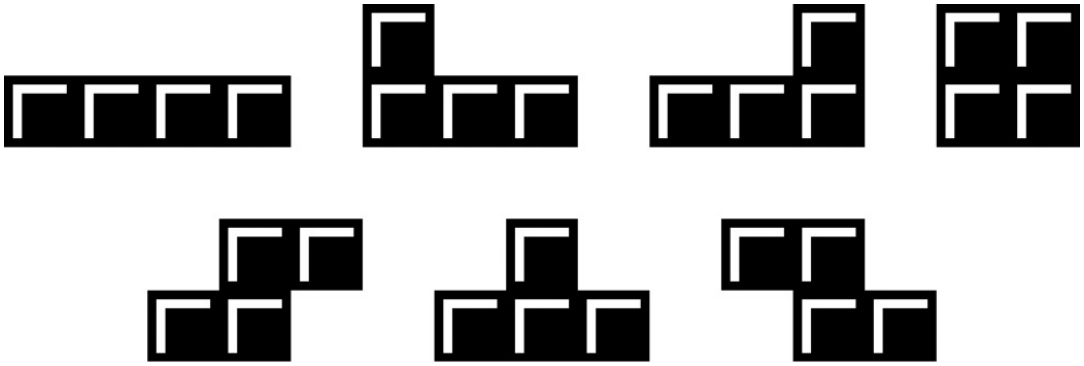
players vulnerable to IT and keeps the game from stagnating. Maybe a player who leaves the tree has temporary immunity to allow her to get safely out of IT's sight, or maybe it becomes a stand-off, where the escaping player has to wait for another player to distract IT's attention before she can make a break for it.

What about freeze tag? In this case, a player who's tagged by IT is "frozen" and has to wait for another player to come and "rescue" her before she can move again. This variation has much more direct interaction between the non-IT players. Instead of just depending on one another as decoys, they have to actively put themselves at risk to aid other players, which only adds to the tension of the game. And it creates a new dynamic between the non-IT players: I rescued you this time, but if I get tagged you're going to have to leave the tree and rescue me.

And that's what games are good at: exploring dynamics, relationships, and systems.

## The Story of Tetris

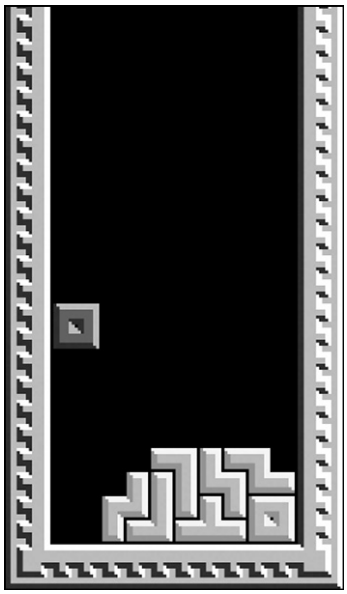
A “system” is what we’ll call the interaction (or ongoing interactions) between a set of rules. Let’s talk about Tetris now.

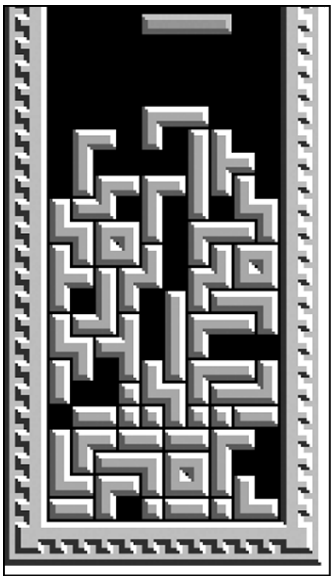


What are the rules of Tetris, essentially? The basic rules that drive Tetris are:

The game is played with *pieces*, comprised of every possible combination of four squares. (See the image above.)

- Pieces fall continuously into a well of a certain volume. The player can guide the pieces’ fall to the left and right of the well, and also rotate the pieces both clockwise and counterclockwise.
- Pieces are removed from the well when and only when the player organizes them into complete rows.
- If there is no room left in the well for a new piece to fall, the player loses.[34]





You can see how these rules create a system where the player's mistakes compound on one another to cause further mistakes: Only full rows are eliminated, so incomplete rows stick around and take up space in the well. Clutter in the well then makes it more difficult to position other pieces and to create rows. As the row fills with mistakes, it eventually becomes impossible to fit more pieces, and the game ends.

These rules function in tandem to give the game a momentum and shape: the player makes errors that cause

further errors, until eventually the player is overcome. (And consider how well a commonly added rule, “the pieces fall faster every time ten lines are made,” works with these basic rules to help the game escalate.) We could consider this a system.

All games aren’t necessarily simulations of existing systems: it would be difficult to imagine a situation in the world that actually resembled Tetris. But it’s easy to imagine simulations that model systems of rules that are far less abstract: urban planning, politics, oil drilling. And there are games whose rules mimic such systems. Will Wright’s *SimCity* is a game in which the player plans a city, Jim Gasperini’s *Hidden Agenda* is a game in which the player governs a post-revolutionary South American nation. Arch D. Robison’s *Seismic Duck*[35] models the way drillers use aimed sound waves and seismogram to find oil reservoirs.

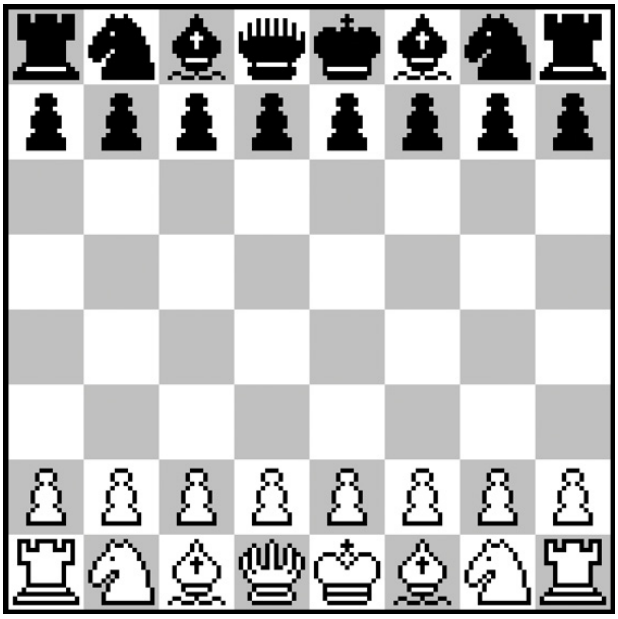
You can begin to see how systems can be translated into game rules: a commercial zone in *SimCity*, for example, needs people to act both as a work force and as consumers. That means the people need homes to live in, transportation to get them around the city, power to make sure the lights are on. The system teaches concepts about the interdependency of urban forces. To again cite Greg Costikyan’s “Maverick Award Speech”: “I want you to imagine a world in which the common person is no longer ignorant of economics, physics and the functioning of the environment—things which are themselves interactive systems—because they have interacted with them in the form of games.”

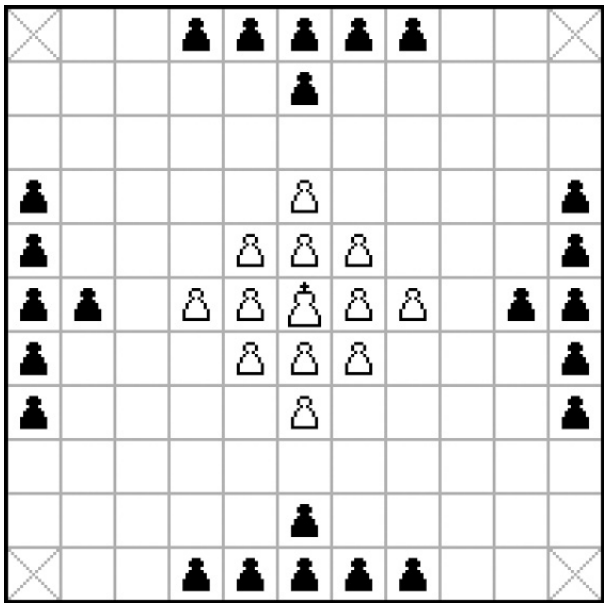
Every game of Tetris has the same shape—errors compound on errors until the well is filled and the player is overcome—because the system of rules we’ve discussed guides the experience in that direction. But the player places all the pieces herself. Every player will place the

pieces differently, will play a different game, but experience a similar result. The same holds true for any system of rules, as simple as Tag or Tetris or as complicated as *SimCity*. Games have a lot of potential for examining the relationships between things—or, rather, for allowing the player to examine the relationships between things, because the player does not merely observe the interactions; she herself engages with the game's systems.

## The Rise of the Designer

Tag is an example of a folk game, along with Go, Chess, Poker, Stickball, Hide and Seek, and most of the world's oldest games. Games have been around as long as civilization has; the game is by no means a new form or a recent invention. What is relatively recent is the shift from folk to authored games. Folk games, like folk songs and folk texts such as the Bible, have no single credited author, but rather many untraceable authors over many years. They're artifacts shaped by entire cultures, and generally they can tell us a lot about those cultures.





For example, compare Chess, a continental European board game of warfare, with Hnefatafl, a Viking board game of warfare. Chess is a game of combat between kings with equal resources. Each player has the same pieces and starts in the same position on opposite sides of the game board. Each player's goal is to capture the other player's king. In Hnefatafl, one player represents a king and his defenders, who start in the center of the game board. The other player represents the attackers, who surround the king's forces on all sides of the board. The king player's goal is to get the king through the attacking hordes to safety, while the other player's goal is to surround and capture the king. The differences between these games' interpretations of combat tell us a lot about the differences between strategic thought between European vassal kings and Viking warrior bands: their priorities, the nature of their battles, and whether they approach warfare as a platonic war between equals. And the games themselves, in turn, shape the strategic thought of those who play them.

Our history is full of folk board games. Authored board games—games created by a single person or small group, and whose authors can be identified—are a more recent

phenomenon. For example, I can tell you that the board game *Cosmic Encounter* was designed in 1977 by Bill Eberle, Bill Norton, Jack Kittredge, and Peter Olotka of Eon Games. (We can date *Cribbage*, by Sir John Suckling, to the 1630s.) These are games as texts of specific rules, rather than as patterns of rules that are subject to change through mimicry. A game of Tag will always have a chasing player and a safe position, but the actual rules will change from play to play. The majority of contemporary board games are designed by a single author or team, and the same is true of digital games.

Can there be folk videogames? Videogames retain credits better than board, card, and physical games. I think that there are digital games, though, that exist as patterns of similar rules, perpetuated through duplication with small mutations. There are a thousand different versions of Tetris, for example, each coded by one of a thousand different authors, and each version with a slightly different set of rules, a slightly different set of numbers, and often (to avoid litigation) a different name. There's a digital game that's commonly known as "the snake game," which began as an arcade game called *Nibbler* by Joseph Ulowetz and John Jaugilas. In this game, the player directs a snake to gobble pieces of food. The snake dies whenever it crashes into either a wall or its own body by coiling around itself. Each piece of food causes the snake's tail to grow longer, making it take up more space and making it more difficult for the player to avoid collisions with her own body. So many different authors have remade this game on so many different machines that all of its forms and variants are usually just referred to as "the snake game." Is this how authored games become folk games?

But what can authored games tell us that's different from folk games? Folk games tell us about the culture that created them; authored games tell us about the author that created them. Authored games have the potential to be

more personal, and thus more specific and diverse, than folk games. Two plays of an authored game are likely to be more similar than two plays of a folk game, because the authored game retains the rules set created by its original designer. It's the fact that folk games change with each player that makes them so long-lived, that makes them adapt to suit the culture that adopts them. But in this book, it's authored games, and the diverse set of voices they embody, that I want to focus on.

## What's Video Good For?

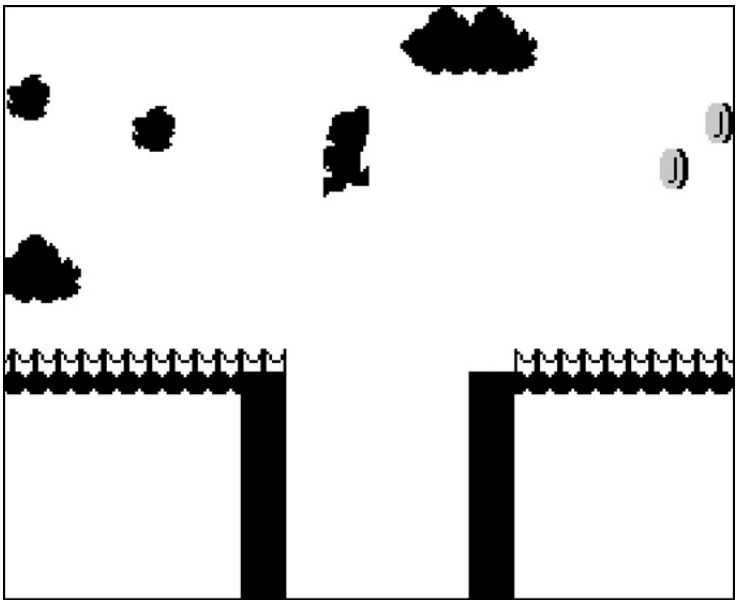
In a board game, players have to track how much money is left in the bank, which pieces are in play, how high the water level rises. A deck of cards can keep players from knowing in what order pieces will come into play, dice can generate random outcomes to situations, and players have hands of cards that represent information they keep from the other players, but beyond these basic devices, little information can be hidden from the players, because the players must make sure the rules are being observed by tracking most of the information themselves.

In digital games, the computer keeps the rules. The computer tracks all the numbers. Digital games therefore have much greater control over what information the players have access to, making videogames capable of much greater ambiguity than board or card games.

What's ambiguity good for? Telling stories! Digital games have great potential for storytelling. The author has a lot of control over the pace at which information is revealed; therefore the author can pace the telling of a story. This is not to say that videogame stories are being told as well as they could be. But the format of a videogame—which lets rules be changed and introduced over the course of the experience, and which lets the author hide the causes for events and show only the effects—lends itself more easily

to an overt, sustained narrative than any physical game format.

Because the rules are kept by the machine, the rules in digital games tend to be more numerous and more subtle. Think of a game like Shigeru Miyamoto and Takashi Tezuka's *Super Mario Bros.* Unless you've studied the game in great detail on a technical level, you probably don't know exactly how high Mario can jump relative to the height of the screen, or how fast he accelerates horizontally when he runs. The interactions between these hidden rules in videogames can result in very complex systems without necessarily complicating the game, because the player isn't required to track and compare all the numbers. For example, imagine the designer creating a situation where there's a tiny platform with a long pit on either side. Mario has to run to build up the momentum to clear the pit and land on the platform, but instead of stopping there he needs to immediately jump again in order to make the second pit without losing the momentum that will let him cross it. This is a problem that wouldn't be obvious to someone who had just approached the game.

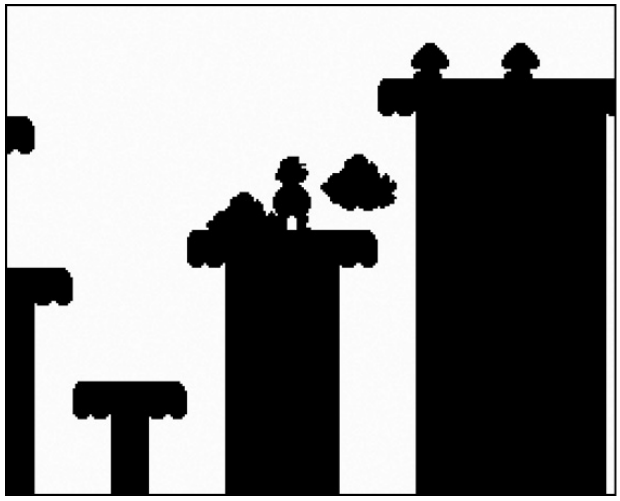


Through playing the game, the player develops a sense of the limits and subtleties of these hidden rules. This interaction between the player and the game, dependent on the game's hiding information, gives digital games their special capacity for subtlety and nuance. You could compare it to the use of "English" in a physical sport: the difference between hitting a ball and hitting it with a particular force, and in a particular direction.

Because of this capacity, videogames are often performative: they allow the player room to interact with rich and complex systems with grace and finesse. We usually refer to this as "skill." A system may persist through an entire game, but the game may start very permissive of less graceful playing and require the player to play with more and more finesse as the game goes on. The game gets HARDER, asking that the player become more skillful, but allowing her to learn the game's systems over the course of navigating increasingly difficult situations.

The systems that the player manipulates in *Super Mario Bros.* are introduced very early in the game, with the only added rules coming with the periodic introduction of new enemy characters or hazards. But the situations that Mario has to navigate start fairly relaxed and demand more and more skillful playing as the game progresses. In the first stage of the game, obstacles are low enough that a simple jump from a standing position will allow Mario to clear them. In later stages, the height of obstacles will require Mario to run and build momentum before jumping, in order to jump higher. In this way the designer teaches the player the subtleties of the game's complex system through careful use of machine-controlled variables. Digital games are thus good at teaching, and at communicating a sense of the player's progress, which often parallels the progress of the protagonist and the development of a story.

What else is handy for telling a story? The ability to generate or play video and audio, either as accompaniments or as central vehicles for information. Digital games can incorporate a variety of media when telling their stories. Consider how the music in *Super Mario Bros.* speeds up when there's only a hundred ticks left on the time limit to complete a stage, creating a sense of urgency, or how the sound played when Mario jumps on an enemy gets higher and higher pitched, indicating that a reward—in this case, an extra life—will come if the player keeps doing what she's doing. Consider how the player's journey takes her through a changing visual landscape, from a sunlit field to a black-and-blue underground, to treetops, to the mushroom forest, and to Bowser's castle, and the way each of these sights—withheld from the player until her skill develops to give her access to later areas—provides a sense of progression through the Mushroom Kingdom.



I don't mean to imply that non-digital games are incapable of the things I've described, or that digital games are in some absolute sense better or more worthy of interest. There are many different kinds of games, all of them suited to different things. Digital games, because of their ability to

withhold and pace the player's access to information, because of the strict narrative control the author is able to have over the player's experience (because the machine enforces the rules), and because of their capacity for generating a wide variety of sights and sounds to enhance or even define the playing-out of the rules, are particularly well suited for the telling of stories. And the telling of stories—games becoming more personal—is what especially interests me about games as a form.

## Role-Playing Games

Digital games have certain strengths for telling stories, but the nature of games in general—even without the advantages the machine provides—makes them good for storytelling. And I do think, in the way that film and photography have generally changed the focus of novels and visual art, the mechanical rules keeper that videogames provide has caused similar focus changes in other aspects of games. In the last chapter, I couldn't discuss the earliest mainframe computer games without mentioning role-playing games like *Dungeons & Dragons*. Role-playing games came out of the tradition of "miniatures wargaming," a set of rules for moving armies of dolls around a tabletop battlefield and pitting them against one another in combat. Later role-playing games kept the rules for combat and situation resolution, gave players the responsibility for a single combatant rather than a larger number of soldiers, and largely got rid of the dolls.[36] But the important thing that games like *Dungeons & Dragons* introduced was the concept of a "Dungeon Master," or "Game Master." This is a player who manages the game for the other players, laying out the scenario and directing the world's responses to the players' actions. The Game Master essentially inhabits the role of storyteller, preparing and

guiding the players through a story in which they make decisions.

This aspect of role-playing games—an overseer who negotiates the player's choices using a set of rules—was eagerly adopted by many of the first digital game authors. Early digital games like *Rogue*, a graphical game of maze exploration and combat, lifted its rules and probability calculations directly from *Dungeons & Dragons*. And text adventure games like *Zork* took the idea of a narrator who relates the world of the game to the player using a consistent voice. Many early digital games are, conceptually, role-playing games in which the computer takes on the role of Game Master.

The computer's adoption of the responsibilities of rules keeping and number counting has shifted the focus of many tabletop games away from their original focus—providing players with an extremely finely grained simulation of combat and other adventuring situations. Why spend thirty minutes rolling dice and looking up random treasure and critical hit information from lengthy tables when you can play a computer game that resolves everything, with far greater mathematical complexity, in seconds and in color? Instead, tabletop role-playing games have been able to move away from a mathematically dense combat situation and toward collaborative storytelling and improvisation.

Take, for example, Paul Czege's *My Life with Master*, in which the players invent a Victor Frankenstein–style mad scientist under whom they will serve as Igors. The Game Master, as the titular Master, assigns the servants duties that enable the players to act out the conflict between their duty to obey and their desire to reclaim their humanity. The dice rolls aren't to test the player's ability to penetrate leather armor with her sword, but rather to test the player's "love" versus her "self-loathing": if a player "fails" her roll, her character must either perform the duty assigned her or refuse to perform it, regardless of which

outcome the player might prefer, a simulation of the state of being in emotional servitude. In the end, one of the servants will rise up and destroy the Master, a result that the rules of the game make inevitable: the destruction of the Master is the climax of the story, which each player finishes in turn by creating an epilogue for her character.

What interests me about *My Life with Master* isn't just its use of rules as a unique device for telling a story of personal, internal conflict, rather than as a means of resolving physics simulations in a fantasy world. I also find Czege's distribution method interesting. *My Life with Master* is sold on Czege's website as both a book, sent in the mail, and as a downloadable PDF file that the buyer may print herself if she's interested. Digital distribution! In the past, the rules for tabletop role-playing games were so elaborate that they required hardbound books, distributed through traditional bookstores and novelty stores, which is still the method used for distributing recent editions of *Dungeons & Dragons*. But authors of small, more experimental role-playing games like *My Life with Master* are avoiding or mitigating the costs of publishing and distributing by selling their rules online as digital downloads, or in some cases simply posting them for free on the Internet.

Role-playing zinesters! And ones who, through their change of focus from complicated and expensive rule books full of encounter tables to simple rules that create conflicts and guide the players in creating a story, offer useful lessons that the designers of digital games could stand to learn.

## Grown-Up Games

Games are useful, I wrote earlier, for exploring and teaching about dynamics and relationships. *Gang Rape* is a role-playing game made by Tobias Wrigstad in 2007.

Outraged by how many rape cases the courts in his native Sweden dismiss without charges, Wrigstad wrote a game that he hoped would allow players to explore and talk about the experience, and the horror, of rape.

Most games are designed to be pleasing and stimulating to play in an immediately rewarding way: they're intended to be fun. This game is not like those games. In fact, the rules—which are only available by directly and personally requesting them from the author[37]—open with the sentence: "A scenario about gang rape is not meant to be fun to play." The game is intended to be harrowing; its goal is to give players some respect for the severity of its subject.

A game isn't defined by being fun just as comics aren't defined by being funny. A game is defined as an experience created by rules. Wrigstad's *Gang Rape* is like any other game in this regard.

In the traditional role-playing games we've discussed, one player takes the role of Game Master. That player then guides the other players' experiences by telling another player, for example, what things her character "sees." *Gang Rape* has no Game Master—one player is a victim, the others are her attackers—but it gives each player limited, Game Master–like control over the characters the other players are playing. Specifically, the rapists can tell the victim how her body reacts to their actions, but not how she feels about those actions. The victim can tell the rapists how they feel while they perform those actions. Rape is about control: these rules are designed to give the players an impression of the power a rapist has over the person being raped. There are additional rules that allow the rapist players to dictate each other's behavior, and to allow the players to explore the role of peer pressure and "egging on" in the dynamic of a gang rape. All the rules are clearly crafted to create a sense of the dynamics at work in a

situation where two or more people have power over another, and to give the players the liberty to explore and better understand those dynamics. (Though the scenario the author had in mind is clearly a woman being raped by two or more men, he admits that the characters involved—and their players—can be of any gender, and that the rules can apply to a scenario like bullying and mobbing instead of sexual assault.)

This game (which, again, is distributed exclusively digitally, and with the special limitation that anyone wishing to play it must identify herself directly to the author) is an example of using the capabilities of games—experiences created by rules—not to indulge an escapist fantasy but rather the direct opposite: to try to educate players about the dynamics at work within a horrible real-life experience, and how those dynamics might come to be as a product of individual choices and responses. The players narrate, through their characters, the events prior to the assault, the events of the assault, and the aftermath.

Games can be topical; they can be relevant to our lives as human beings. They can be relevant without having to be about rape. But *Gang Rape* is one example of what we gain when people other than commercial publishers author, publish, and distribute games that commercial publishers could never touch.

## The World's a Stage and We are Players

Often, games—particularly digital games, with their use of video and audio—are compared to film, probably because the videogame publishing industry strongly resembles the Hollywood studio system. But I don't think this comparison is particularly constructive, in that it gives us little insight into what the game, as a form, is capable of. Film tells a static story; what's exciting about the game is that it allows the audience to interact with a set of rules. This doesn't

mean the game can't tell a story: in the role-playing genre, the players aren't merely watching a story but playing the roles of the characters within the story.

A better comparison than film is theater, which is where a lot of our game vocabulary ("the player," "stages," "set pieces," "scripting") comes from. A play defines the roles, events, and scenes of a story. An individual performance of those roles and scenes will always be different: different actors will perform the same role in different ways. Every performance and interpretation of a particular play is different—sometimes in minute ways, sometimes in radical ways—but we consider the play itself and the scene itself to be the same.

Compare this to a game story, particularly a videogame story. Every player will perform the story called *Super Mario Bros.* differently (and the same player will perform the story differently each play), but the role of Mario and the actions Mario is capable of taking remain the same. There is always a scene called "World 1-2," although each performance of "World 1-2" will be different. In a more contemporary videogame such as *Half-Life 2*, a very clearly cinema-inspired game, each player will always pass through the events the designers have scripted in the order in which they are presented, but each player's (and each play's) performance of Gordon Freeman, the game's protagonist, will be at least subtly different. The player will always get chased across the rooftops by cops, but in one performance she might hesitate, unsure of where to go, in one she might head straight for the escape route, in one she might panic, almost getting Gordon Freeman killed, and in another she might walk a little too close to the edge of the roof, fall, and have to start the scene over.

As game storytellers, we are not directing static stories take-by-take but rather arranging the scenes that will comprise the shape of our story. We can begin to think of the player as someone performing a role we've written

rather than as an audience who experiences our story without any input as to its outcome. We allow room for improvisation, room for the player to make a role her own. The audience of a game can be more usefully compared to the audience for a play than the audience in the movie theater. In videogames, the audience is there, live, with the actors—or as the actors—experiencing a single performance that is unique, despite the story having been performed and continuing to be performed many times.

Some players record videos of their performances, either for documentation or for the purpose of recording a specific achievement, such as reaching the game's conclusion as quickly as possible—what is usually called a "speed run" (YouTube has given lots of these videos a means of reaching an audience). That there's an incentive to capture individual performances of a game testifies to the amount of variance there is within a game depending on who's playing it.

## Games and Chance

The board and card game traditions have also given a lot to digital games. What I think digital games have taken the most from board games and card games is the way they manage chance. Both contemporary designed games and older folk games have invented many systems for managing chance. The six-sided die, for example, allows for the random selection of six equally likely outcomes (and can then be further used to access other percentages and ratios; for example, three outcomes, each represented by two sides of the die, or eleven outcomes with different likelihoods represented by two rolls of the die, and so on.)

Card games themselves are designed as a system for managing chance and gradually revealing information. When all cards are in the deck, every card in the game has (as far as the player knows) an equal chance of being in any position. Once a card has come into play and been seen by the players, though, the players then know where it is and can use that information to make guesses about the remaining cards. Cards also allow players to manage the pace at which they reveal information: a player might have a hand of seven cards hidden from the other players, who don't know whether those cards have come into the game yet or not. Poker is a classic game of using limited knowledge of the cards in play to predict the positions of cards not yet in play. This is what makes Poker an elaborate game of bluffing. One player tries to see through the other's "Poker face" because the decisions she'll make are based on what she can predict about the information the other player is concealing. Contemporary game designers have contrived even more rules to control the revelation of information.

Aside from hiding information, chance is frequently used to break symmetry. Having different starting conditions

between players prevents both players from having the same set of ideal moves, and thus having the game become a stalemate. Having different, randomly selected values between one play and another, or having different game events happen at different, impossible-to-predict times (or not at all), means that each game will demand a different strategy, keeping play from becoming stagnant.

Franz-Benno Delonge's and Thomas Ewert's board game *Container*, for example—a game where players trade and transport commodities—uses chance to ensure that all players do not value the commodities identically. At the start of the game, a number of cards are shuffled and randomly distributed, one to a player. These cards describe how valuable the different commodities are to the players who hold them, and each card values the commodities differently. The cards are also kept hidden until the end of the game, each card seen only by the player that holds it. Because each player is aware of the entire possible set of values on the cards—she knows which cards are in the game, and which card she, and therefore not the others, possesses—she can watch the other players' decisions and make deductions about which players have which cards, and therefore which commodities are valuable to which players.

Computers have an innate capacity for manipulating chance. Though true randomness doesn't exist, computers handle numbers easily and are capable of generating reasonably unpredictable probabilities of any size on the fly. Every computer has access to an infinite number of monkeys rolling an infinite number of dice.

Why is this useful? Because, as we've discussed, games have a unique capacity for improvisation! Though each scene has the same shape—Link battles a gang of Moblins—each performance is different. So what if, in one performance, one of the Moblins comes from the left instead of the right? Digital games have the capacity to

create variations on many subtle details in every play, keeping the experience from becoming stagnant.

The differences don't have to be subtle, either. In Chris Klimas and Joel Haddock's online game, *Where We Remain*,[38] for example, the player is a boy searching for a girl on an island patrolled by monsters that are intended to evoke characters from Greek mythology. The layout of the island—what tools are hidden in which caves, what areas which monsters patrol, and in which cave the girl is hidden—is different every time, decided by a random number generator. In effect, this randomness makes the characters and events of the game more archetypal because the emphasis is on the shape of the game—the boy's search for the girl while monsters pursue him—rather than on the details like what treasure is hidden where. Games have lots of room for improvisation, for every play of a game or scene to be unique, and digital games in particular have easy access to a great degree of chance.

## Games as Culture

I keep bringing up the profound influence role-playing games like *Dungeons & Dragons* have had on digital games. Both Eastern and Western videogame trends have their roots in *Dungeons & Dragons*, but both experienced and responded to that influence in different ways, much as Chess and Hnefatafl reflect different experiences of and responses to warfare.

Character creation is an important feature of role-playing games: the players literally describe the role they intend to play, both in narrative terms (what is this character's background and personality?) and in mathematical ones (how many times can this character be hit by dragon breath before she collapses?).

The American game *Wizardry*, created in 1981 by Andrew Greenberg and Robert Woodhead, was an attempt to bring the dungeon exploring and monster battling of *Dungeons & Dragons* to the computer. (It has plenty of peers with similar intentions and similar properties, but it's a good example for discussion.) In *Wizardry*, the player controls a team of up to six adventurers, similar to a team of player characters in *D&D*. Each of these adventurers is, fundamentally, a set of statistics (Strength, I.Q., Piety, Vitality, Agility, Luck, Age), a Class (the character's job or specialization), and a list of Spells (tactical magical abilities characters can use to help with combat or exploration). Character creation is held over from tabletop role-playing games, but in *Wizardry* it loses its narrative dimension. Players "roll" statistics (as with dice) and assign character classes and ability sets based on those random "stats," and that is the extent of characterization.

Character creation is present in many Western digital role-playing games, though implemented in different ways. The *Ultima* games, which I've mentioned before, ask the

player a series of moral choices that determine which character class she plays as. Games like Michael Toy and Glenn Wichman's *Rogue*, which borrows its combat rules directly from *D&D*, skip character creation but assign the player's character no properties beyond this set of stats.

Yuji Horii's *Dragon Quest*[39] is similar to early Western digital role-playing games: character creation is limited to entering a name, which also determines your stats via a hidden algorithm. *Dragon Quest* was widely imitated in Japan, in addition to receiving many of its own sequels, but over time Japanese games broke with the mathematical focus of character creation in Western games. The trend Japanese digital role-playing games tend to follow is to have the player character be increasingly designed by the game designer. In the *Final Fantasy* series, characters wear outfits designed by the authors, answer to names chosen by the authors (though the player is sometimes given the choice of changing these names), and speak scripted dialogue that the player has little say in.

Compare a 2002 Japanese game like *Kingdom Hearts*, a collaboration between the authors of *Final Fantasy* and the Disney corporation, to a 2008 American game like *Fallout 3*, produced by Bethesda Game Studios. In *Fallout 3*, the player not only names her character, but also designs her face and appearance and decides on her race and gender. In *Kingdom Hearts*, the player plays a character named Sora, who is given clear motivations by the writers and dialogue by voice actor Haley Joel Osment and whose name cannot be changed. Sora's appearance in *Kingdom Hearts* is mostly static: he changes appearance to fit some of the worlds he visits, but the only part of him the player is allowed to change is what his weapon looks like when he's swinging it. His outfits, designed by the game's art staff, are law, as is his race and gender: white and male.

Why has character creation remained such a fixture of American interpretations of digital role-playing games while Japanese role-playing games have phased it out? It could possibly reflect that America is a young country, and a nation that has been capitalist almost since inception. American culture sells the idea of individuality and ego. In Japan, a much older country in which social roles are valued (and connected to uniforms), role-playing might more easily mean playing the role to which you've been assigned. (In Yuji Horii's *Dragon Quest*, the protagonist's sole characterization is that his ancestor is a hero.) There's an ongoing dialogue between Eastern and Western design these days, so none of these trends are exclusive (and they've never ruled all of design, obviously), but there are clear patterns in games that we can trace to the values of the people who created them.

Games tell stories that communicate the values of their creators in a unique way: not just through their explicit content but through the logic of their design, and the systems they choose to model. And if games communicate the values of their creators in a unique way, then it's absolutely essential that there be more creators passing on more values, more perspectives. Games must become more personal.

## Footnotes

33 Sometimes it is: consider the contemporary interest in game theory among non-game designers.

34 This list obviously omits some rules, like how the game selects the pieces to bring into play next, and at what speed the pieces will fall. Digital games tend to have many more of these very specific rules.

35 http://home.comcast.net/~arch.robison/seismic_duck.html.

36 It's worth mentioning, by the way, that labeling traditions of games the predecessors of digital games is not to imply that these traditions are obsolete and gone. I know people with armies of tiny soldiers.

37 You can contact the author to request a copy at http://jeepen.org/games/gr.

38 Playable online at http://twofoldsecret.com/games/whereweremain.

39 The *Dragon Quest* games appeared in America under the name *Dragon Warrior* until recently, for legal reasons: Horii borrowed the name *Dragon Quest* from a specific *Dungeons & Dragons* supplement.

# Chapter Four Changing the Game

Creating a game from scratch takes time, effort, and tools—tools that haven't started to become widely available until recently. Digital games are capable of presenting video and audio, and creating a game often requires that music be composed, sounds be recorded, art be drawn, or characters be animated. That involves a variety of tools and training and a lot of work. Often it requires a team of people, each with a different specialty, to accomplish a particular design.

And while plenty of hobbyist authors work on that level, the truth is that most digital games are not made from scratch. Most tell their stories by borrowing pictures or sounds or music or code. They use kits that other people have made; they use sound effects that other people have recorded and released for their use.

And many, many game creators, budding or otherwise, piggyback on existing works, taking advantage of existing infrastructure to make their creations. A commercial game from a large studio, for example, requires thousands of 3-D models, sounds, and assets to be loaded onto the player's computer. Even more important, a commercial game comes with game logic: a means of resolving collision, of determining the effects of gravity, of moving objects around a game world. These things are difficult for someone inexperienced to code, but here a group of paid professionals have already done all the work. The player who's bought this game has all of those resources on her computer: why not use them?

In this chapter we're going to talk about what are usually called "hacks" and "mods" (for "modifications"): changing existing games to create new stories. There are a variety of ways in which people change games, and a variety of

motivations for doing so. Sometimes changes are akin to crude vandalism, such as redrawing Super Mario so he has a dangling naked cock that flops around as the player walks through the Mushroom Kingdom. Sometimes they're akin to clever, subversive vandalism, such as changing the player character into the Princess and the prisoner who waits at the end into the Plumber. And sometimes they're something entirely new, almost indistinguishable from the games that gave birth to them.

Sometimes authors want to situate their creations in the context of a particular work. For example, Jesse Petrilla created *Quest for Al-Qa'eda: The Hunt for Bin Laden* in response to the September 11, 2001, attack on the World Trade Center, which the American media attributed to Osama Bin Laden. This game is a modification of *Duke Nukem 3D*, a first-person game about ultraviolent American culture in which the stripper-tipping, alien-gunning protagonist appears to the player only as two attributes: a gun that hovers in front of the camera and a series of prerecorded movie quips. The jockish American attitude of military vigilantism totally fits the play of *Duke Nukem*, a game in which the most logical response to everything the player sees is either to shoot it or to stuff bills between its tits. It makes sense that *Quest for Al-Qa'eda* and Petrilla's 2003 sequel, *Quest for Saddam*, should take the structure of that game.

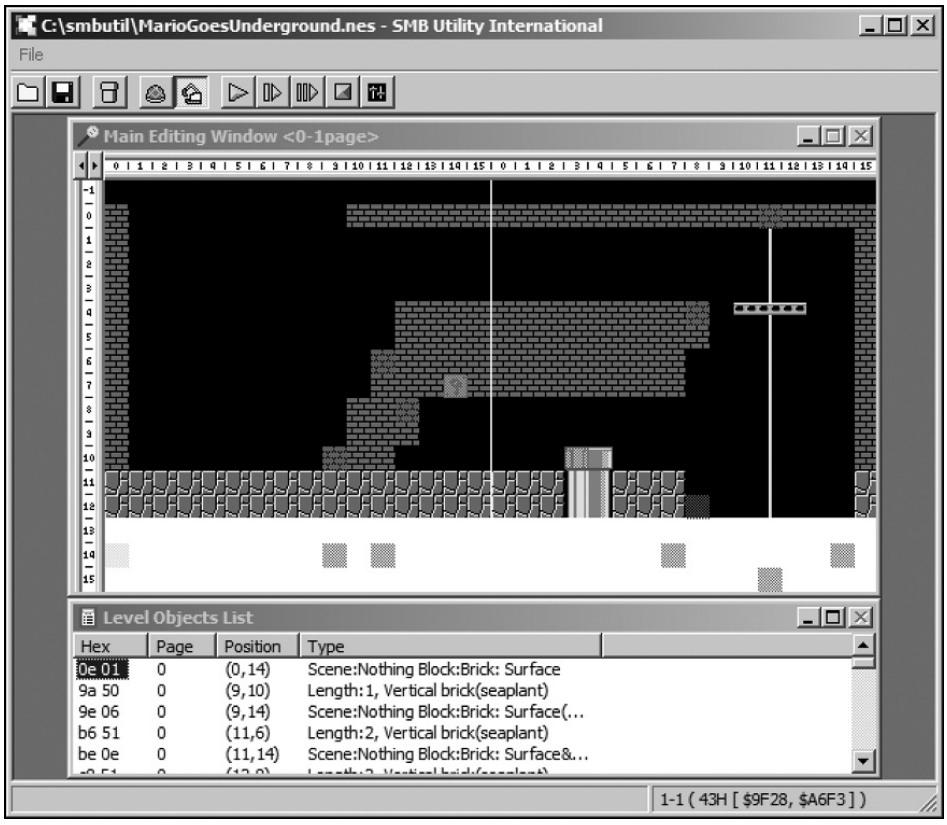
It further makes sense that the Global Islamic Media Front should, in 2006, reinvent *Quest for Saddam* as *Quest for Bush* (or *The Night of Bush Capturing*), a modification of *Quest for Saddam* in which the Saddam Husseins the player ceaselessly guns down have been replaced with George W. Bushes, an icon of American militarism. They're structurally the same game, with the faces swapped and the pictures on the wall changed in the simplest possible inversion of the original game's xenophobic aggression—a

blunt “how do *you* like it?”[40] Hacks and modifications are made to subvert or comment on the original author’s intentions, or to simply correct what the modder feels is an oversight on the author’s part, or simply because it’s an easy existing infrastructure for creating something new. Let’s examine the varied motivations for hacking a game by looking at the most-hacked digital game: *Super Mario Bros*.

## Another Castle

*Super Mario Bros.* was published on a cartridge that plugs into the Nintendo Entertainment System. Hardware has existed for a while, however, that allows the contents of a Nintendo cartridge to be "dumped" to a computer and distributed digitally as software. The digital file is called a "ROM," and "emulator" programs have been written for many different machines that will run that game software as a Nintendo Entertainment System would. In this way, *Super Mario Bros.* has made the transition from hard media to a purely digital form, available for modification by anyone with the knowledge and tools.

And tools have been readily available for a long time. *Super Mario Bros.* was written in Assembly, the machine language I mentioned in chapter 2, and reprogramming the game requires working knowledge of Assembly. But programmers cracked the code a long time ago, and plenty of tools currently exist for changing the appearance, rules and levels of *Super Mario Bros*. without having to look at machine code. When I made my 2008 *Super Mario Bros*. hack, *Mario Goes Underground*, I used a tool called SMB Utility[41] that made it easy to rearrange *Super Mario Bros.*'s maps.




*Super Mario Bros.* has enjoyed popularity with ROM hackers for a variety of reasons. For one, the large number of people who've played the game made it an early target for programmers to crack, and tools have existed for a while. It's also widely played enough that most people have a sense for how the game works: move left, move right, jump, catch a mushroom to grow big. Most players have an idea of what to expect from *Super Mario Bros.*, so they can pick up a hack easily and be surprised effectively when the hack turns out to be different from the original.

Some hacks simply change the game in a superficial way. *Silhouette Bros.*, by Leon Arnott, changes all the characters and architecture to stark black silhouettes on a solid-color background. The hack demonstrates how recognizable the elements of *Super Mario Bros*. are and the choices of color are well suited to the tones of different parts of the game. *Enigmario* by Dr. Floppy replaces the soundtrack of the game with covers of songs by the band Enigma. The

somber tones of this new soundtrack give *Super Mario Bros*. a more introspective feel.

*Mario vs. Airman* is an interesting attempt at intertextuality: the "Airman" Mario fights is a villain from Capcom's *Mega Man 2*, and Mario is required to navigate a level from this game to defeat Airman. Mega Man, as a player character, moves very differently than Mario. Momentum does not affect his motion, he can "brake" instantly, and his arm gun makes him able to shoot his opponents, rather than requiring him to jump on them to defeat them. Trying to navigate a level designed for Mega Man as Mario is a challenge that emphasizes the importance of the nuances of motion in platform game design.

A hack like *Super Mario Forever* exists to challenge the player's technical knowledge of *Super Mario Bros.*'s rules. Mario starts mid-fall into a bottomless pit, and must instantly maneuver to land on a tiny platform. The game then requires the player to generate enough momentum on that tiny platform to make a long jump to the next, which involves jumping off of and back onto the platform in order to build speed.

Some hacks change a single rule of a familiar game in order to give the game a new experience, similar to the concept of "house rules" in non-digital games. Normally, the player of *Super Mario Bros*. must complete each stage within a limited time, which resets at the stage's end. Nanashi's *900 Time Challenge* gives the player a single time limit in which to complete the entire game: nine hundred ticks of the game's internal clock. Completing the challenge requires the player to minimize the wasted time in her play, and to exploit shortcuts and warp zones in order to get through the game in as fast a time as possible.

Then there are hacks like *Extra Mario Bros*. (or my own hack, *Mario Goes Underground*), which attempt to tell an entirely new story using the familiar framework of *Super*

*Mario Bros*. Rather than requiring Mario to proceed from the starting point on the left to the castle on the right, *Extra Mario Bros*. contains a big world for the player to explore in multiple directions, using pipes to travel vertically from area to area. It's essentially *Super Mario Bros.* through the lens of contemporary game design, which is why it's interesting as a direct modification of *Super Mario Bros.*: the games that informed *Extra Mario Bros*. are all descended from *Super Mario Bros*.

Then there's *Super Daisy Land*, which changes the star of the Nintendo Game Boy game *Super Mario Land* from Mario to Daisy, a captured princess who normally waits at the end of the game for rescue. But that's not exactly a hack of *Super Mario Bros.*, and there's a richer subject for gender correction in game mods.

## Bobs Whose First Name is Betty


Anne-Marie Schleiner, artist and writer, maintains Mutation .fem, a gallery of modifications of first-person shooting games: games where the player's character is personified

as the gun she holds in front of her, like the earlier-mentioned *Duke Nukem 3D*. Each mod changes the gender of the characters in those games. The patches hosted on Mutation.fem[42] change male characters to female for a variety of reasons: either for the purposes of gender play, to correct gender bias by the games' authors, or to subvert the aggressive masculinity of this school of game creation.

The first mod in Mutation.fem that I played was Lynn Forest's *FemDOOM*. The original *DOOM*, a 1993 release from id Software, was one of the earliest games to popularize the first-person shooter, and has given tropes to that genre that are still imitated today. The game is viewed from a first person perspective, and the protagonist is given no name or background beyond being a space marine assigned to a tour of duty on Mars. Only the tiny, emoting face of a man in the status bar and the occasional baritone grunts when the player is shot gender the protagonist as male, and reinforce the tacit assumption that the player of a game of aggression must be male.

Lynn Forest, a fan and player of *DOOM*, was frustrated with the implication that she was playing a game for men. So she drew femme faces to replace the male faces that appear in the status bar, and put her patch online for other players to download. A later patch, "FEMDMSND.WAD," credited to "Amanda, Ivor B. and Rob Lord," replaces the masculine sound effects with feminine ones of equal intensity.

Gender is even more of an oversight in id Software's later *Quake*, which puts an even greater emphasis on competitive play. In *Quake*, one sees other players not as masked figures, but as hulks with a single male face. Players modded the game to introduce feminine faces and bodies. Schleiner notes[43] that later versions of *Quake* acknowledge players' desire for female avatars by including female models. She also hosts "skins" worn by

PMS Clan, the Psychotic Men Slayers,[44] a band of women players who play competitively together in the game *Quake 2*.

Mutation.fem contains patches that paint a mustache on *Tomb Raider*'s protagonist Lara Croft and change Bungie's *Marathon* from a game about firing guns in dark, angular space stations to a game about fighting with egg flippers and dish towels in a checkerboard-tiled kitchen. Loren Petrich's "Tina-Bob" patch replaces the generic jumpsuited men—the "Bobs"—that solely populate *Marathon*'s space station with identically jumpsuited women.

## The Hack as Sampling

The repurposing of commercial game assets can be compared to sampling in music: using part of an existing song as an instrument in your own piece of music. Hip-hop artists often rap their own material over music sampled from another source, using the found music as a background for their own words.

*HyperBound* was created in 2009 by Michael Iantorno as part of his thesis project at Ryerson University.[45] Iantorno's game is a hack of *EarthBound*, a Super Nintendo Entertainment System game developed by APE and the Japanese writer Shigesato Itoi and released in 1994. *EarthBound* is a digital role-playing game informed by Yuji Horii's *Dragon Quest*, but instead of telling the story of a warrior who battles wolfmen and dragons in a fantasy world, *EarthBound* is set in a world vaguely resembling a Japanese vision of contemporary America. The protagonist is a young boy in a baseball cap who fights aliens and renegade animals using an old baseball bat and his newly discovered psychic abilities.

*HyperBound* takes its name from "hypertext," text that's arranged in a nonlinear structure. (This book is a text: it's arranged to be read from start to finish, one page to the next. A website, where you might click on a word to "link" to a page about that subject, is hypertext.) What better model for the nonlinear exploration of text than the space of a digital game, where the player moves around the world by moving her character across the screen, encountering characters, and listening to what they have to say? That's the part of the design of *EarthBound* that *HyperBound* has lifted. What it's rejected is the fighting. The hack is purely about exploring the world and discovering the text, an original script written by Iantorno and his brother.

In *EarthBound*, the protagonist typically wears a baseball cap, a striped shirt, and a backpack. There's a brief scene at the beginning of the game, though, where he appears in pajamas with ruffled bed-hair. That means that there were animated sprites of the bed-headed protagonist, seen from all angles, just waiting in the game for Iantorno to use. *HyperBound* uses those sprites for its protagonist, to give him an identity apart from that of *EarthBound*'s protagonist and to emphasize his confused state: the protagonist is amnesiac, and the information he is trying to recover is knowledge of his own identity.

Throughout the hack, Iantorno repurposes assets from *EarthBound* to fit his new story. The bearded, sunglasses-wearing criminal the player encounters in *EarthBound* becomes the radio DJ whose show the protagonist of *HyperBound* used to call in to before he lost his memory. The boarding school that appears in *EarthBound*, with its classrooms and lockers, becomes the university that *HyperBound*'s protagonist attended, where he meets former teachers and finds valuable information on his previous life. All of *EarthBound*'s graphics are sampled and given new purposes in the hacked game.

The other assets Iantorno samples are less obvious, though just as crucial: the ways in which the player presses controller buttons to move the protagonist around a world, to engage characters who wander around of their own accord in conversation, and to advance their dialogue text on the screen. These are all things that are non-trivial to program, and that have been programmed for Iantorno by APE. To present his nonlinear story, he's taken advantage of an existing infrastructure for allowing the player to navigate a world of characters who can be spoken to.


If you can create a script but you can't create animated characters, scenery, or code to take player input and move a character around a large world that scrolls in eight directions, why not use those existing assets to present your original script? The work has already been done and the resources already exist to be sampled.

## Changing the Script

There’s another obvious reason to change the script of a game: to translate it to a new language. Publishers often neglect to translate and publish a game in a new language because of the cost: if they think a game won’t sell enough in a part of the world to justify paying people to rewrite the script, to manufacture cartridges for different game-playing hardware, and to market the game in another country, they won’t do it. But dumped roms, which are purely digital, don’t require the expense of physical publishing, and because they’re distributed freely, don’t depend so heavily on marketing to make up that expense. And thus, people who read languages neglected by videogame publishers and care enough about games from other parts of the world will invest their own time and effort into the unpaid translation of games.

The work of translating a game involves more than just translating the script—which is often a lot of work, given that the dynamic nature of games usually means lots of scenes and incidental text to translate. The new script also needs to be inserted into the original game. In formats where space is expensive, such as Nintendo game cartridges for example, programmers use a lot of clever tricks to store text as efficiently as possible. Cracking those codes can be tricky. And if the original game was written in Japanese, the hacker also has to insert a new font. The way the game displays text might not be suited for English letters, so screen space is usually at a premium, too. Japanese is a compact, ideographic language, and a word that takes up two Japanese characters on screen could be an eight-letter word in English, or an idiom that requires translation as a full sentence. How do you translate the word so it fits? All game text tends to be presented in a limited part of the screen (a window that takes up the

bottom half of the screen, for example, so the player can see the characters who are speaking on the top half), so all word choice is informed by just how much space is available to display those words—space that was chosen based on the structure of an entirely different language!

The Japanese Game Boy Advance game *Mother 3*, a sequel to *EarthBound*, was translated into English over the course of two years by a two-man team who refer to themselves as "Jeffman" and "Tomato."[46] The script for the game was written by the Japanese writer and journalist Shigesato Itoi. Nintendo's American branch decided that translating the game would be too much work, and too expensive a project, given the limited audience they expected the game to have. Nine years before, Nintendo of America put a lot of money into translating *Mother 2* into *EarthBound* and manufacturing copies, only to have their marketing department completely mishandle it. Physical distribution is a big investment, after all: Nintendo of America was unwilling to make the investment a second time. American players who had encountered and liked *EarthBound*, though, were eager to play the sequel and were upset that it wasn't to be published in English. Tomato, who works as a professional Japanese-to-English translator, played the Japanese game and decided that English-reading players deserved to experience the game. He undertook the translation of the game, and Jeffman did most of the coding to get around the memory limits of the original Japanese game: a two-year project, undertaken for free. The English version of *Mother 3* was released in July 2009. You can go on the official translation site's forums to read how many players cried during the ending[47] (which is, incidentally, Stephen Spielberg's metric for when games will become art,[48] not that tears are anywhere close to the only metric for judging the value of an experience).


But don't be misled into thinking that the majority of translation work is in translating Japanese games to English. As of this writing, people are working to translate *Mother 3* into Spanish and Latin American Spanish, Portuguese and Brazilian Portuguese, French, German, Italian, Dutch, and Malay. Whenever I visit the ROM hacking database at romhacking.net, I see translations of English games into German, Polish, Russian, Korean. Hackers are engaged in correcting the oversights of profit-oriented thinking and making the games they care about available in their native tongues.

## Machinima

The product of “machinima”—from *machine* and *cinema*—isn’t games, but the source material is. It’s the same kind of sampling I’ve described: machinima uses the resources and infrastructure of commercial games as the basis for creating animated movies. All the resources are there in most 3-D games: the ability to move a camera through a world full of 3-D objects, and to move and animate those objects. Game hacking provides an immediate avenue to 3-D animation, one that’s far more accessible than dedicated 3-D modeling (with a specialized—and costly—modeling program like Maya or 3D Studio Max) because there are already scripts in place to move things around and to operate a camera.

The origins of the community surrounding machinima can be traced to that of the game *Quake*, a fully 3-D first-person shooting game that offers players the ability to record, distribute, and play back “demo” recordings of their games. Players began to manipulate and position the camera during these playbacks, introducing cinematography to the demos. Soon they were directing story scenes to frame their demos.

Most machinima stars the characters of the games they sample, because that’s what’s available: usually armored soldiers who crack jokes for the benefit of players familiar with the game. The 1996 *Diary of a Camper* by Matthew Van Sickler, Heath Brown, and company,[49] for example, follows a band of *Quake* characters as they deal with an opponent who is hiding and picking off their comrades one-by-one. (And who is ultimately revealed to be John Romero, one of the founders of id Software.) *Diary of a Camper* is widely remembered as the first game demo interested purely in storytelling, rather than presenting a sample of play footage.

But the subject matter of machinima has become increasingly diverse. The 2003 machinima *Anna*, produced by Katherine Anna Kang's (wife of *Quake* programmer John Carmack) Fountainhead Entertainment, follows the life of a single flower on a forest floor.[50] *Anna* bears little resemblance to *Quake* at all, set in the woods with a cast of plants and animals.

As more tools become available, machinima looks less and less like videogames. A few years ago I encountered a movie of Sherwood Anderson's short story "The Dumb Man" filmed in *Second Life*,[51] a game that gives players the tools to construct anything they like, be it movie sets or actors or other games.

While machinima is not games, it's demonstrative of the ways that game tools can be used to create things radically different than the originals: including new games altogether.

## Something Borrowed

Few commercial games, these days, are made entirely of original resources. The creators of the *Quake* and *Unreal* "engines," the rules and code that drive the games, license their engines to other developers so that they can save time and work by modifying an existing framework rather than building an entirely new one. The *id Tech 3* engine—the engine that runs the third *Quake* game—was used to create games like *American McGee's Alice*, *Call of Duty*, *Star Trek: Voyager—Elite Force*, and *Star Wars Jedi Knight II: Jedi Outcast*.

The publishers of these games paid for licenses allowing them to sell and distribute their modifications, but plenty of players, using their knowledge of how their favorite games work, changed the games into something new and distributed their modifications for free.

Brandon Chung's *Citizen Abel: Gravity Bone*[52] was built in a modified *Quake 2* engine, but the finished game bears little resemblance to a game about space marines blowing up aliens. *Gravity Bone* is a spy pastiche that opens with the player riding a fancy gated elevator into a masked costume party. Once there she must make contact with a fellow agent. This sets forth a rapidly accelerating chain of events that lead to an unexpected outcome.

What *Gravity Bone* takes from *Quake* is logic for moving a player around a three-dimensional world, looking around with a first-person camera, and collecting and manipulating items. Everything else—the game's appearance and presentation, the locations and scenario, the player's goals and the events they trigger—is original, designed and inserted by Brandon Chung. *Quake*, here, is just a vehicle for the delivery of Chung's original story.

Or look at Robert Yang's *Radiator* series,[53] which is built in the *Half-Life 2* engine. The second episode of the series,

*Handle With Care*, involves the player visiting a marriage counselor. What *Half-Life 2* does well is allow the player to manipulate and move objects using a rudimentary physics simulation: in *Handle With Care*, the protagonist, Jason, withdraws to a “Repression Facility” while his husband speaks with the counselor. The Repression Facility looks like the kind of grimy steel structure common to *Half-Life 2*. In the facility, the player carries crates containing repressed memories (if there’s one thing *Half-Life 2* does well, it’s letting the player carry crates around). If they’re put on the facility’s shelves, they’re locked away forever; if they’re broken, Jason revisits the memory he was trying to repress. The therapy session, seen through a monitor, progresses depending on whether Jason confronts or seals away his memories.


Both of these games look significantly different than the games they’re based on. Nevertheless, we can identify rules they’ve borrowed from their mother games. It’s apparent why it’s easier to change existing games into new games rather than creating them from scratch: particularly when it comes to 3-D games, solitary creators often don’t have the resources that the programmers of *Half-Life 2*

had. It’s unlikely that either game could’ve existed without such sampling—it’s far less likely that Valve would make a game about the roles individual memories and pasts play in negotiating modern queer relationships than a *Half-Life 2* level involving building a wall of crates to stop drone planes from killing you.

Scripting cameras and collision, as well as creating 3-D architecture and a way of storing it conveniently in computer memory, are all big projects. The Valve Hammer Editor, which edits *Half-Life 2* levels, gives a creator immediate access to all the design tools a team of paid programmers put into place. It also gives her access to the resources *Half-Life*’s art team created, which would otherwise be another huge investment of time, effort, and skill. Editors like Hammer allow creators to create without having to build an infrastructure to manage their creations.

## New Worlds

Many game designers have anticipated their players' desires and created editors to accompany their games. The first level editor I ever encountered was for the Nintendo game *Excitebike*. *Excitebike* is a stunt bike race. The included EDIT MODE allows players to mix and match the obstacles from the game—ramps, bumps, gravel pits, and boost pads—to create new stunt tracks to race on. When I was little, I spent a lot of time with the instruction book open in my lap to the list of pieces, moving my racer through the track and rearranging the obstacles. The appeal of an editor like this to a child who's creative but lacks experience is that everything—the rules, the code, all of the art and sounds—are provided. The only thing I had to worry about, as a kid, was the track itself.

When we were discussing shareware, I mentioned Tim Sweeney, who founded Epic MegaGames—which later became Epic Games, the publisher of the *Unreal* engine used by many commercial and independent designers to build 3-D worlds. The first game Tim Sweeney sold, as "Potomac Computer Systems," was released in 1991 and is called *ZZT*. (He picked this name so it would always be the very last game listed on alphabetically ordered shareware CDs.) The shareware part of *ZZT* was a series of four self-contained adventures, the first of which came with the game and the other three of which had to be purchased. (They were later released for free.) The free part of *ZZT*, however, was the editor that Sweeney used to make those adventures, and the means to play "worlds" created with it.

*ZZT* is a text-mode game: it takes advantage of DOS computers' native ability to display text. In addition to letters and numbers, DOS has a special extended sheet of characters for drawing basic pictures: smiley faces, playing card suits, Greek letters, and a variety of borders, lines,

and patterns. *ZZT* co-opts these: the player is a smiley face, Greek characters such as pi and omega are monsters who roam around, the diamond from the playing card suits is a gem that the player can collect and use to buy items. This set of 255 characters comprises everything that any *ZZT* world will ever contain.


And that's incredibly liberating. It means that *ZZT* is self-contained: there's no need to go outside the editor to find another tool to draw and animate graphics to import into the game. Authors get 255 characters in sixteen colors. And creative authors can do a lot with those: in that set there are patterns of varying density and lines of varying thickness, and I've seen some pretty impressive environments and portraits drawn with those text characters.

Sound and music are similarly self-contained: *ZZT* uses the PC speaker, which contains a range of simple notes and instruments. *ZZT* has a simple scripting system for playing PC speaker tunes, which again obviates the need to go outside the world editor for anything.

But oh, yes: let's talk about scripting. *ZZT* has a nice library of game pieces, which appear in Tim Sweeney's shareware *ZZT* worlds. There are pushable boulders,

“rotators” that move objects around and can form conveyor belts if a series of them is placed in a line, stock monsters, weapons, and power-ups, and several kinds of walls (invisible walls, secret passages, walls that can be shot and destroyed). But the most interesting game piece is the Object, a programmable creature that can take on any appearance. Each Object contains a script written in a language called ZZT-OOP (ZZT Object-Oriented Programming), a simple programming language.

Here’s an example of ZZT-OOP:

```
/n/n/n/n/n/n
#shoot e
Haha, I shot your favorite PRICELESS VASE!
```

That tells an Object to move north (or up) six times, then shoot a bullet east, then display a haughty message on the screen. That’s pretty simple, something someone who’s never coded before should be able to understand. Though it’s simple, ZZT-OOP is also surprisingly robust: each Object has its own script, and can receive messages from the player and from each other. Here’s a more complicated example of two Objects interacting:

```
@Priceless Vase
#end
:shot
#send Collector: broken
#die
```

```
@Collector
#end
:broken
You broke my favorite PRICELESS VASE!
#char 1
/i
#char 2
/i
#char 1
/i
#char 2
```

When the vase is shot (the colon indicates a message the Object can receive) it sends a message to the object named Collector (the @ indicates an Object's name), which causes the Collector to display a message and briefly animate (#char is the "change character command"; character 1 is a hollow smiley face and character 2 is a filled smiley face. The direction "i" means "idle"; it tells the object to wait a moment between flipping its character).

Authors have taken advantage of ZZT-OOP to do things author Tim Sweeney never envisioned. One can create a *Super Mario Bros.*–style jumping game by creating button Objects that, when touched by the player, send messages causing a Super Mario Object to move left or right or jump, and that falls when there's nothing beneath it. Recently, someone built a machine that graphs sine waves in *ZZT*. Creative authors have bent the tools they were provided with to entirely different ends than they were intended for.

What's interesting about *ZZT* is not only the robustness and versatility of ZZT-OOP, but the simplicity of the game's overall presentation. Because all of the graphics and sounds that a *ZZT* world can produce are already inside it,

it frees authors from having to worry about asset creation. They can just design with the tools they have in front of them. This makes it an ideal introduction to scripting and design, and I know a handful of game designers who got their start making *ZZT* worlds.

## House Rules

There's a concept in non-digital games called "house rules." Because players keep the rules in board and card and physical games, and because all the players have to agree on what rules they'll play by, it's easy for the players to change the rules or invent new rules to suit themselves. Maybe they feel as though their modification will correct an oversight on the part of the designer: part of the game may not quite work as it's presented. Or perhaps they've thought of a variation on the rules that's more interesting to them as veterans of the game. Maybe they need the game to be easier or harder, or to change it so it better suits the circumstances in which they're playing it: a game of Tag where the goals change to suit the space that's available, for example.

In digital games, the rules are kept by the computer and thus are somewhat hidden from the player. Adding house rules to a digital game seems more far-fetched. There are plenty of ways, though, to change games: changing the rules, inventing new rules, inventing new scenarios and whole new games out of the same content. Telling new stories!

What's really valuable about hacking and modifying games is the realization that there are ways of interacting with games other than just playing them: roles beyond consumer. Inventing rules is, after all, inventing games. The knowledge that games can be changed and remixed is the knowledge that games can be created: it's a small conceptual leap from the one to the other. While game creation tends to be more interdisciplinary—using the Valve Hammer Editor allows the creator access to all of the 3-D models, textures, sounds, characters, and such from *Half-Life*, letting her focus on design—there are game creation

tools now that are as easy to approach as the typical level editor.

I said that the first level editor I encountered was *Excitebike*'s. I've played with plenty through the years: *ZZT* and its sort-of successor *MegaZeux*, *Dezaemon 3D* (a spaceship shooter kit for the Super Famicom), *Knytt Stories* (a jumping-and-climbing story creator), *Tombs of ASCIIroth* (a text-mode puzzle game kit), *Ragdoll Masters* (rubber dolls fighting each other in the air), *The Elder Scrolls IV: Oblivion* (a fantasy computer role-playing game). I've used whatever tools I could get my hands on to tell stories.

You can, too. And once you've realized that videogames—even if they're made by big corporations with teams of hundreds and budgets of millions—are mutable, and can be reinvented by a single person, you can start to imagine what's possible when you have the means to create games that are entirely your own.

## Footnotes

40 More information on the history of all three games can be found at http://www.gameology.org/reviews/quest_for_bush_quest_for_saddam_content_vs_context.

41 http://www.zophar.net/utilities/neslevel/smb-utility.html.

42 http://www.opensorcery.net/mutation.

43 See "The Female Skin Pac for Quake": http://www.opensorcery.net/mutation/patches.html.

44 Newman, James. *Videogames,* 57. The PMS Clan still exists, but has apparently been co-opted by corporate interests and has been rebranded Pandora's Mighty Soldiers.

45 http://hyperbound.net.

46 http://mother3.fobby.net/or..

47 Though you probably shouldn't. Spoilers.

48 Lev Grossman, "The Art of the Virtual," *Time,* November 8, 2004, http://www.time.com/time/magazine/article/0,9171,995582,00.html.

49 http://www.youtube.com/watch?v=uSGZOuD3kCU.

50 http://www.youtube.com/watch?v=bKEr5RRKoO4.

51 View it online at http://vimeo.com/609147.

52 http://www.blendogames.com.

53 http://www.radiator.debacle.us.