

PUTTING THE GAME AND LEARNING TOGETHER

It's a matter of integration.

— Michael Allen, Allen Communications

The “art” of creating Digital Game-Based Learning is integrating the game and learning portions so that the result feels like a fun game and gets the learning accomplished. It is worth reemphasizing that there is no “cookbook” solution as to how to best do this. Rather, this is the place where the highest amount of creative thinking is needed.

Categories of Digital Learning Games: Different Means for Different Ends

Learning games can be categorized in a number of different ways. All of these categorizations are worth considering when deciding how to integrate your proposed game or game style with your content. They include:

- Intrinsic vs. extrinsic games
- Hard-wired games vs. “engines” and “templates” or “shells”
- Tightly linked games vs. loosely linked games
- Reflective games vs. action games
- Synchronous (real-time) games vs. asynchronous (turn-based) games
- Single-player vs. two-player vs. multiplayer vs. massively multiplayer games
- Session-based games vs. “persistent-state” games
- Video-based games vs. animation-based games.

Intrinsic vs. Extrinsic Games. In 1970, Tom Malone, just out of Stanford and working at Xerox PARC, published a landmark paper entitled “Towards a Theory of Intrinsic Motivation.”¹³ In it, Malone made the argument that there are two main categories of learning games: *intrinsic* and *extrinsic*. In an intrinsic game, Malone argued, the content is an integral part of the game structure. His example is a math game, where things go up as you get to higher quantities and down as you get to lower quantities. A more contemporary example is a flight simulation game, in which the game itself is about flying a plane, or *Sim City*, where you learn

the rules of urban development by trying and succeeding or failing. Most simulation-type games fall into this category.

Extrinsic games, on the other hand, are games where the content and the game structure are less tightly linked, or not linked at all. The paradigm here is the question or trivia game in which the questions can be about any subject, but the game remains essentially the same. *Bingo*, *Jeopardy!*, and other often-used training games fit into the extrinsic model.

Which model, intrinsic or extrinsic, is better? Proponents of each will give you reasons why theirs is superior, and this is actually a highly controversial topic among Digital Game-Based Learning designers. “Intrinsic games,” says Michael Allen of Allen Communications, “may provide the most powerful learning experiences technology can support. These are perhaps the most noble and worthwhile applications of technology in the learning field.”¹⁴ Clark Aldrich argues that “the real power is when you capture the rules at an algorithm level and have people understand them through constant exposure to different circumstances.”¹⁵ On the other hand, anyone who’s ever used a typing game knows it can be fun *and* can help you to learn. I believe that *both* intrinsic and extrinsic games have their value in different situations. The tradeoff you need to think about is that while intrinsic games enhance certain kinds of learning and add to the engagement, they are typically created on a custom basis and are therefore more costly and often difficult to change or update. Extrinsic games, while lacking the learning power that may come from tightly integrating the content into the game, lend themselves well to templating and to rapid changes of content, often at lower cost. Remember that intrinsic/extrinsic is not an either/or proposition; it is a continuum. There are a number of states between the two, one of which I refer to as *loosely linked*.

Tightly Linked Games vs. Loosely Linked Games. This categorization of learning games is somewhat similar to the intrinsic/extrinsic classification, but is actually a different perspective. A tightly linked game is one that is constructed specifically around a fixed set of content. The content is built into the game; knowing the content is vital to succeeding in and winning the game. A tightly linked game can still be extrinsic, and the entire game might be able, with a fair amount of effort, to be repurposed for other content. A detective game in which the clues

are pieces of information about the product is an example of a tightly linked game.

A loosely linked game, on the other hand, is one in which the content is essentially separate from the game, but there are “hooks” in the game which bring the two together, and send the player from the game to the content, and back again. In repurposing the game to new content, only the hooks must be changed, not the whole game. *The Monkey Wrench Conspiracy* is an example of a loosely linked game. It is a task-based learning game in which the tasks—which are done outside the game in the software to be learned—are initiated by encountering flashing objects, which, although part of the story line, are easily changed to add, eliminate, or alter a task.

Like extrinsic games, loosely linked games often allow content to be changed much more easily than tightly linked games. That means you would use them in situations where, say, the content was still in development, or changing rapidly. A tightly linked game is better for incorporating unchanging content; for example, the fixed model in the game *Situational Leadership*.

Hard-Wired Games vs. “Engines” and “Templates” or “Shells.” The ultimate tightly linked game is the so-called hard-wired game. Here, the designers and programmers sit down with the goal of building only this particular game. Reusability is not a consideration. *Everything* in the game is designed and optimized around the game, the content, and the player experience. In many ways, if done well, this will produce the best game of all, just as a custom-tailored suit is likely to look and fit better. But it is a very expensive way to do things.

The opposite of the hard-wired game is the template, or shell. In this approach, the content, be it text, graphics, video clips, or whatever, sits somewhere external to the game, is “read-in” or “called” by the program at the appropriate time, and is displayed onscreen. This allows the construction of content-editor software, in which a trainer or teacher can just type in various pieces of the content and have the content automatically displayed in the correct place in the game.

An approach in between hard-wiring and pure templates to use is what is referred to by programmers as an “engine.” An engine can be, for example the software that lets you run around a three-dimensional world

realistically, not walking through walls and encountering objects and things that move and have various properties. Such an engine can underlie or “drive” equally well a shoot-em-up game such as *Doom*, *Quake*, or *Unreal*, or a nonviolent, more politically correct game such as *Straight Shooter!* A number of game engines are available commercially; game companies often amortize the large expense of developing them by licensing them to other companies for other games. The *Doom*, *Quake*, and *Unreal* engines are all on the market (typically at high prices for commercial use), as are many individually developed versions.

Training vendors often take the trouble to turn what were originally custom-developed, hard-wired games into engines, so that they can resell them in a number of different contexts (this is also referred to as *templating a game*.) In this case, the *interactions* in the game make up the engine, and all the graphics and words change according to the new context. Examples of this are *Time Out!*, which was converted from being about a manufacturing company to being about a financial company, and *Running a Hotel* where the same engine was used for games about running a phone company branch office and running an elevator company.

The least happy result of all comes when a game is hard-wired because the designers or programmers are inexperienced with games and just plunge ahead building it as they go without considering reusability. This can happen either because they don’t think they have to make things reusable, or because they don’t know how to, or both. Hard-wiring a game should be avoided as much as possible in a final product. Prototypes, however, are often built hard-wired because doing so is faster and cheaper.

Reflective Games vs. Action Games. As we saw in the last chapter, there are a number of genres of games, ranging from action to role playing to strategy. A differentiating characteristic between these types of games that has an important bearing on Digital Game-Based Learning design is the degree of reflection they allow, because this is an important part of the learning process that is often under-included. Nonstop action games (also known as *twitch games*), offer the least opportunity for reflection in themselves, while role-playing, adventure, simulation, strategy, and puzzle games often proceed at a slower pace and offer more built-in reflective “space.” (There are, nevertheless, twitch puzzles, such as *Tetris* and *Devil Dice*, as well as less reflective real-time strategy games.)

Role-playing games typically let you make choices in various types of dialogs, which provide reflection points. Adventure games, where you go around and find objects that allow you to solve puzzles, also give time for reflection around “how do I solve this—what do I need?” Simulations games, such as running-a-company sims, often allow you to make decisions at your leisure, although some provide real-time time constraints; strategy and so-called “god” games often give you all the time in the world to make up and change your mind.

Does this mean that we can or should never use a twitch game as part of Digital Game-Based Learning? No, it does not. The important thing is that there be a good balance of action and reflection in the final product, just as there should be a good balance of *edu* and *tainment*. Too much action and there’s no time to reflect. Too much reflection and it can become boring. Again, we need to find the “flow path” between the two. This is part of the principle of *pacing*, which is so important to novels, movies, games, and other devices meant to hold our attention.

Synchronous (Real-Time) Games vs. Asynchronous (Turn-Based) Games. The distinction between real-time (also known as *synchronous*) and turn-based (also known as *asynchronous*) games is quite important to Digital Game-Based Learning, in at least two ways. In single-player mode, a real-time game must be “paused” or put into a pause state to interrupt it, either for reflection or to do something else. This usually involves saving the “game state” (everything that is happening) at that point. Some real-time games—for example virtual pets—do not allow this. The game continues whether or not you are playing; stop playing long enough and you lose. In a turn-based game, on the other hand—chess for example, but also many strategy games—the machine will wait forever for you to figure out your next move, unless you’re “playing by the clock.”

The distinction between synchronous and asynchronous is even more important in multiplayer games. While a game in which everyone is playing with or against each other at the same time—for example, a real-time battle or a competitive business simulation—can be very interesting, in training it can usually only work when trainees are in the same situation at the same time, such as in a training class. But this is often not the case in online training, at which point turn-based games, which allow

each player to play whenever he or she has the time, may be a better solution. A turn-based game, though, may lack some of the immediate excitement of a real-time game, so the engagement has to be produced in other ways, such as a real interest in the outcome as in *President 96*.¹⁶

Single-Player vs. Two-Player vs. Multiplayer vs. Massively Multiplayer Games. Games can be either single player, multiplayer on the same computer (some *You Don’t Know Jack* games), two or multiplayer over a network or the Internet, or massively multiplayer, which means that hundreds, thousands or potentially even millions (although not today) can play either at once, or on an “in and out” basis.

Most Digital Game-Based Learning to date has been single player, except in the military, where the goal has always been to link people because that’s how war is fought. As discussed earlier, an issue for multiplayer games in business is getting the people together. In consumer games, this is often done via a virtual lobby where you first go when you want to play. When enough people are there for a game, the game starts. Some games allow people to join while the game is in progress. Another option for large multiplayer games is the persistent state game.

Session-Based Games vs. Persistent-State Games. Session-based games exist only for as long as the initial players are playing. Although the game may be able to be paused and resumed, it pauses for all players at the same time. When somebody wins, the game is over, and must be played again. In persistent-state games, on the other hand, the world of the game never goes away—each player moves in and out of it as they wish, but like the real world, the game world continues. In persistent-state games, such as *Ultima Online* or *EverQuest* (persistent-state games are often role-playing games), you can build up skill and experience over time, which has obvious implications for learning. Because the world is always continuing, and opportunities may come and go, there can also be penalties for not playing, and this, too, has clear connections to real business life. An example of a massively multiplayer, persistent-state business game is *Star Peace* from Monte Cristo.¹⁷

Video-Based Games vs. Animation-Based Games. Another interesting choice that designers of Digital Game-Based Learning need to make is whether whatever representations of characters and places they

include will be video or animation based. These are two very different schools, often reflecting whether the designers have a video background. The advantage of video is absolute realism. Its disadvantages include the size of the assets (they are large and may limit what can be put on a CD or sent over the Internet) and limits on interactivity (because the scenes need to be prerecorded so they can play out when they are required by the player's choices.) Both of these disadvantages have been overcome to some extent by proponents; the first by better compression and streaming methods, and the second by techniques for cutting videos into pieces as short as 1 or 2 seconds, and assembling video sequences on the fly. A good example of this is *Angel Five* (see Chapter 9). An additional issue to consider with video is that if any changes need to be made, the video many need to be reshot, necessitating reassembling the same cast, sets, lighting conditions, and so on. This may lead to difficulties, as Video Arts found out when they wanted to shoot additional scenes for their older titles only to find the actors, such as John Cleese, had aged considerably.¹⁸

Animated characters and graphics, on the other hand, allow designers a great deal of freedom. They can be made to look and sound any way you want. Unlike live actors, they never age or become unavailable. Their behaviors and actions can be preprogrammed and reprogrammed as necessary. "If I want a character to storm out," says Richard Berkey, creator of *Strategy Co-Pilot*, "I can just hit the 'storm out' key."¹⁹ Animated characters and graphics are less expensive and, of course are totally lacking in ego (although that's not necessarily true of their creators).

Which to use in a Digital Game-Based Learning project depends on several factors. First, a need for absolute realism might suggest using video, although animated characters are fast approaching video actors in the details of what they can do (their voices, which are recorded actors, have always been real; computer-generated voices are currently only good for robots.) On today's advanced gaming consoles, the players in sports games appear almost as real as live TV. Second, in thinking about how much realism you really do need, there is an interesting tradeoff between specificity and universality that is explained very well in Scott McCloud's useful book *Understanding Comics*.²⁰ As McCloud shows, the more abstract a character is (the most abstract face being a circle) the more easily we can identify ourselves with that character. As they get more and more photorealistic, characters take on identities that are

increasingly specific and become more difficult for us to project ourselves onto. So, in some cases video, which is *totally* photorealistic may actually hinder player identification with a character. This may or may not be important in a specific instance. If a game's perspective is first-person (through your eyes) and you never see yourself, then it may not matter. If it is "over the shoulder" like *Tomb Raider*, where you do see yourself continually as you play, it may be worth considering.

Richard Berkey of Imparta says that he is planning to move his future Digital Game-Based Learning games quickly from video to animation. In addition to speeding up game creation, he finds that video improves game play, because so many more possibilities can fit on the same CD.²¹ Ashley Lipson, creator of *Objection!*, makes a similar argument: "My product is animation based," he says. "My competitor's product is video-based. My game has thousands of potential paths, theirs has one."²² But Ed Heinbockel of Visual Purple (creators of *Angel Five*) feels that his company's technique of cutting the video into very small chunks and "latching" them produces similar results that are more lifelike.²³

Narrative-Based Games vs. Reflex-Based Games. Another interesting question is how much story to include in the game. Should it be like a movie with an "inciting incident" at the start that makes you want to see the conclusion and complex character development along the way? Or should it be a series of unconnected scenarios or interactions in a gamelike context? The answer depends highly on what you are trying to accomplish. The more you want to create something that is long-term and that builds, the more story is a useful motivator. An alternative, of course, is to have lots of ever-increasing levels of puzzle difficulty, as in *Tetris*.

Although narrative and characters can add emotional impact to a game, which may aid in the recall of certain content, there are also categories of content where not recall but *reflex*—that is, the ability to react very quickly to a stimulus—is what is important. Language learning is one example (How are you? Fine.). Legal objection is another. Acronyms are a third. For such content, reflex-based games, in which stimuli are presented rapidly by the computer (with or without a story-based context) and responses are judged and timed, can often provide an effective, fun, Digital Game-Based Learning solution.

