Rhythm and Cues:
Project Management Tactics for UX in Game Design

Rudy McDaniel, University of Central Florida, Orlando, FL, USA

Joseph R. Fanfarelli, University of Central Florida, Orlando, FL, USA

ABSTRACT

This essay situates game design and development within the domain of sociotechnical research and reviews the results of a case focusing on the design and development of an original video game level with platformer mechanics. Using a case history methodology with autoethnographic methods, the work studies the context in which small game components are authored and methods by which knowledge is exchanged and applied within rapidly developed software systems. It argues that the designer experience is a critical phenomenon to understand within the study of user experience in video games given the iterative nature of development and the necessity of frequent, in-house playtesting. The video game was designed by the authors and developed using preexisting assets from prior projects. Results suggest ideas for incorporating UX strategies into micro-project management techniques that are useful for small and large projects alike. The work closes by calling for future areas of research in related areas.

Keywords: Applied Research, Game Design, Game Development, Player Experience, Project Management

DESIGNER EXPERIENCE AS USER EXPERIENCE

Recent research highlights the importance of gamification for UX and considers how users are persuaded through psychological, physiological, and cultural design tactics (Luoto et al., 2014). However, relatively little is known about the underlying project management techniques from which these tactics are invented and shaped. This is particularly true in experimental video games that are designed by small teams, often with small budgets, for particular types of knowledge exchange. What are the *rhythms* of project management in game development, and how do those rhythms influence the *cues* provided to players? To answer such questions, the *designer experience* can lend insight to our understanding of the *user experience*, particularly in helping us to understand designers' values and the resulting design decisions based on those values. Indeed, since playtesting is such a critical part of designing and developing games (Fullerton, 2008;

DOI: 10.4018/IJSKD.2015070102

Salen & Zimmerman, 2004), the *original* user experience of any game is often a *design team's* user experience while playing through iterations of that game during development.

In the entertainment industry, the development of video games is a complicated affair. Modern "AAA" titles, or those games published by major publishers such as Nintendo, Ubisoft, and Electronic Arts, are developed by large teams with multimillion dollar budgets. Integrating structured and standardized processes to manage such projects is critical for the development of these complex software programs. Further complicating matters is the fact that such titles are designed by employees from diverse professional backgrounds (e.g., art, production, and programming). Many of these employees work in hybrid teams responsible for different parts of the overall game, each part adding to the game as a dynamic system of interrelated components (Hunicke, LeBlanc, & Zubek, 2004).

Within this type of chaotic design environment, traditional techniques for managing complex projects, such as requirements engineering, can support video game development (Callele, Neufeld, & Schneider, 2005). Because games are so specialized, however, we cannot accurately identify one single approach that is universally used by game development companies. While agile development using the Scrum methodology (Keith, 2010), is one of the most popular approaches, it is far from universally adopted and has a number of shortcomings and associated complexities (Miller, 2008). How, then, can we learn more about other methods that have proven fruitful? One approach to gaining this specialized knowledge is interviewing past game development teams or reading postmortem reports, learning from various studios' successes and failures in managing their teams and deliverables.

Unfortunately, acquiring access to teams of commercial video game designers is quite challenging (McDaniel, 2015) and postmortem reports are not always readily accessible to those outside the games industry. Of course, even if such access were possible and such materials were abundantly available, their usefulness would be limited. Large development teams are likely to use processes that work for *their* large teams and *their* specific type of game. Chandler (2010) agrees, noting that the processes that a development team implements need to be customized for a particular project; they should depend on the type of game, the constraints, and the available resources. It follows that video games and UX researchers can learn best from prior works by considering the project management of video games that most closely resemble their own interactive projects.

In this article, we start with a modest goal. We examine the project management of a small video game built by two people, the authors. We call this approach *micro-project management.* Unlike large commercial video games that are birthed from massive stockpiles of documentation including Game Design Documents (GDDs), story bibles, and character sheets, these small independent games are generally less formally planned and produced with fewer employees. In this particular case, we focus on the examination of processes used in the rapid development of a flexible, experimental testbed game. We identify challenges and successes from the design stage of this project; these findings contribute to a better understanding of project management from the perspective of smaller game projects. This topic is relevant to professionals working both inside and outside the realm of game development. For instance, many other types of digital projects are also built by iterating through builds of smaller components that are constructed by small teams of individuals.

GAME DESIGN AS SOCIOTECHNICAL PRACTICE

Sawyer (2013) outlines an approach to studying information systems using sociotechnical theory. This approach considers information systems through their relationships to people, their actions, and the contextual environment in which both people and systems function. Tuffley (2009) further argues that software can be considered as a social system with a technical implementation; along these lines, we can think about video games as offering new opportunities for playful human experiences through technology. Allen and Kim (2003) consider the sociotechnical tradition of video games by contemplating how technology ideas stabilize in certain ways. Following such ideas, it is natural to think about how we might evaluate and assess the user experience within the parameters of stability and remediation, or the re-appropriation of content from one form of media to another. For example, we might consider how enjoyable social groups find a virtual game in comparison to the social groups playing a similar game in the physical world. However, oftentimes new interactive systems do not have reasonably similar analog counterparts. In situations such as these, other methods of evaluation are more useful.

For instance, we can consider the types of testing done with other consumer products to gain an idea of possible methods for evaluating usability or enjoyment. With designed products, techniques such as naturalistic observation have proven useful for better understanding the context in which technical systems are used by people in their natural environments. Early industrial designers such as Henry Dreyfuss (1955/2012) paved the way for modern user analysis through his naturalistic observations of consumers. Working in the 1950s, Dreyfuss designed products ranging from sewing machines to hearing aids and hotel rooms; to assess user preference and product quality, consumer testing was of the utmost importance.

While the type of user experience Dreyfuss tests is critical, there is an additional side of user experience, which is *the experience of design and development*. During development, project management tools and techniques are similarly useful in illustrating aspects of the sociotechnical nature of production. These processes and procedures show those characteristics of production that are most valued by different employees (generally, saving time and improving quality) as well as the ways in which designers and developers interact with one another. Since video games are often complex, built by teams, and composed of complex multimedia assets, the study of video game project management is an especially interesting process to consider within the realm of sociotechnical analysis. This complexity and systems-orientation makes games useful vehicles for showcasing the underlying sociocultural values of their designers and teaching about complex topics like knowledge management (Chua, 2005).

Perhaps even more striking in terms of sociotechnical context is the extreme interdisciplinarity of video game design; engineers must work with artists, for example, to develop aesthetically pleasing levels that still render quickly and play smoothly. This interdisciplinarity is often taxing for project managers. In what Norman (2013) calls the "design challenge," he notes the many complexities of designing in such collaborative, multidisciplinary settings. He explains, "Great design requires great designers, but that isn't enough: it also requires great management, because the hardest part of producing a product is coordinating all the many, separate disciplines, each with different goals and priorities" (p. 34). This is very true in the world of game design and development, where disciplinary preferences may introduce obstacles which must be negotiated in the interest of moving toward viable (and playable) product deliverables.

Prior literature explores project management for video game development, but literature focused on the project management practices of small game development is scarce. Much of this work considers managing projects for complex titles, such as those developed by large commercial studios. Hight and Novak's (2007) work analyzes games project management at large, Wolz and

Pulimood (2007) consider the process from a pedagogical orientation, and Spaulding II's (2009) book emphasizes the roles of teamwork on game production. Techniques and tactics often differ significantly due to the unique nature of production work that is geared toward particular types of gameplay mechanics and specialized game genres.

For example, the terminology, documentation, and methodologies of game development vary from one video game project to another (Chandler, 2010). Further, some documentation practices which appear to be useful at the conceptual level are less so at the practical, "nuts and bolts" level of getting things done. GDDs in particular have been criticized "for not responding to the fluid nature of game design" (Sansone, 2014, p. 113) suggesting a need for considering other types of project management documentation that are more facile and flexible in nature. A broader complicating factor is that there is no one accepted industry standard for the documentation format used during the creation of video games, GDD or otherwise (Greene & Palmer, 2014; Rouse III, 2005). It is a useful exercise, then, to consider specific examples of documentation and design that provide information about how this often mysterious practice is actually accomplished in real world scenarios.

METHODOLOGY

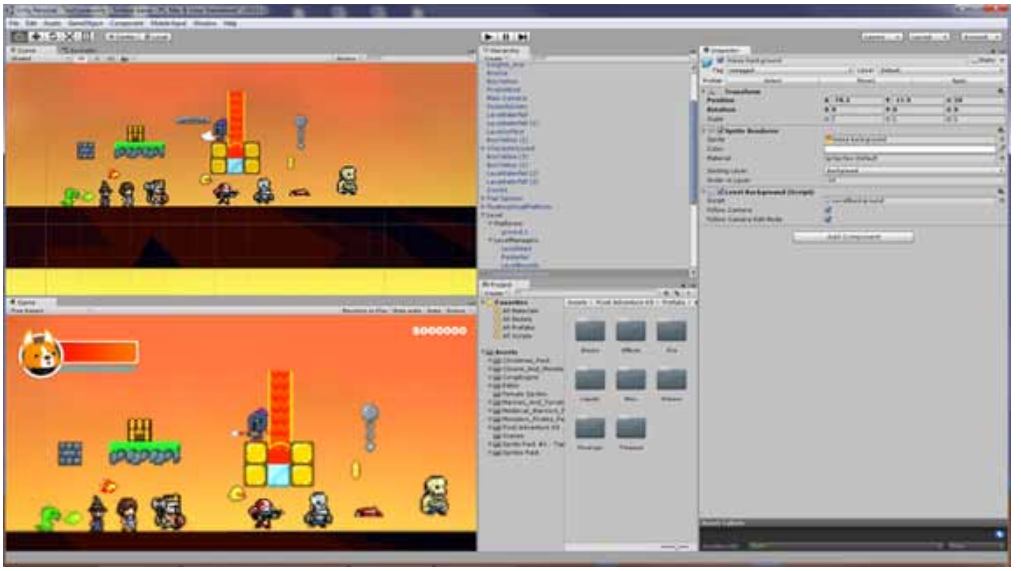
This essay employs a single-case, case history methodology (Yin, 2014) paired with autoethnographic methods (Ellis & Bochner, 2000) to investigate the following exploratory research questions:

1. What project management strategies are useful in the design and development of small, game-based projects?
2. What do these strategies reveal about the user experience of designing and developing games?

Yin (2014) notes that there is some overlap between case studies and case histories, but that case studies are done when "the relevant behaviors cannot be manipulated" and involves both "direct observation of events being studied and interviews of the persons involved in the events" (pp. 8-9). Given our own relationships to the design and development process and the lack of any outside participants, our methodology was not a straightforward case study. However, we did consider a case that was rich with different sources of information and provided documents, artifacts, and observations for analysis, so our methodology blended aspects of autoethnography with the consideration of a historical, though recent, case. A number of studies in the social sciences have effectively used this ethnographic technique of self-observation, including studies documenting practices such as Anthony Wallace's "cognitive mazeway" for driving to work and David Sudnow's process for gradually acquiring the skill to play jazz piano (Anderson, 2006, p. 376).

Given that our primary interest in this essay is the relationship between designer experience and video game project management, we used a variety of different tools and strategies to document the design and development process of our level. Sources of the data used for analysis included our own project notes, sketches that resulted from early face-to-face meetings, and chat logs and virtual sketches made from online tools. We organized these materials into two primary categories: (1) idea generation, and (2) design/development of deliverables. After we grouped these materials by theme, we then selected exemplar cases from each category and used these as points of discussion within the paper. Parts of these exemplars were included as screen

Figure 1. The game used an assortment of premade assets within Unity



shots within the article to demonstrate how they functioned as tools for project management and knowledge sharing during the design and development process.

This autoethnographic case history is presented in two phases. Phase one considers the idea formation and project planning slice of the production process. Central in the design were modular "rhythm groups," or "non-overlapping sets of level components" (Smith, Cha, & Whitehead, 2008) used to increase the efficiency of gameplay and obstacle design. This database of desired user behavior patterns and a corresponding digital drawing tool for synchronous level design enabled us to manage and share of up-to-date knowledge on the game's current state and provided us with components we could later use for further game development. Phase two then explains how the project was developed and includes a discussion of production, testing, and maintenance, from conceptual design to applied development.

CONTEXTUALIZING THE CASE

To investigate the relationship between UX, sociotechnical context, and the project management of independent video games, we consider the project management process of an original game level we developed ourselves. We developed this level using the Unity software suite and built it relatively quickly, over a period of several weeks. As a result, it relies heavily on preexisting assets and scripts (Figure 1), many of which were authored and sold by other developers on the Internet. The game is a two-dimensional platformer; its core gameplay mechanics include activities such as running, jumping, collecting, and fighting enemies.

The purpose of this game is to provide players with an enjoyable experience while also allowing us to evaluate gameplay patterns and collect data from the players on how various game elements incentivize different types of in-game behaviors. For example, one key item of interest within the game is the impact of achievements on player behavior. As a result, the level is designed with a number of digital badges, or achievements, that reward the player for doing

certain things within the level. If a goal were to examine the effect of badging on play time, for example, then a badge could be awarded upon a player's earning of a large number of coins. The total number of coins could only be earned by exploring every nook and cranny of the game level in order to accumulate enough virtual tokens.

PHASE 1: IDEA FORMATION

Initial Planning and Purpose

Our primary design goal was to develop a game that could be manipulated for experimental research purposes. Since there were only two people involved in the construction of the game, much of our initial planning was done informally, through face-to-face conversations in one of our offices, often with accompanying sketches made on pieces of paper or even napkins. Through frequent casual discussions, we identified a series of different experiments we could design. We considered identifying a series of contrasting games for each experimental manipulation, but that seemed inefficient.

Ultimately, we decided upon a different concept. One of the researchers had recently developed a video game focused on teaching different areas of brain function and successfully used it in experimentation (Fanfarelli, 2014). Perhaps, instead of identifying several pairs of games, each for a single experimental manipulation, we could develop an experimental testbed game for conducting experiments on game design features. While creating a game from scratch is no simple task, a game designed to be flexible lends itself well to multiple experiments. Also, being able to add and remove game design implementations quickly allows the game to be re-used with minimal modification, providing ongoing longitudinal benefits. In other words, designing such a game, while taxing at first, ultimately reduces our workload by providing a useful system for game design experimentation for the foreseeable future.

Moreover, this game could be designed specifically for experimental purposes. Not only would it be a game that could be modified in terms of experimental manipulations, but it could also be modified to track and record data. In this way, we could further reduce our workload by reducing our reliance on external assessments that need to be combined with play data that has been tracked through vigilant observation of gameplay. Thus, our task was set. Our main design goals were as follows: first, we needed to have the ability to create levels quickly and reconfigure them quickly, and second, we wanted to have custom tracking abilities for assessment and evaluation purposes so that we could quantify gameplay through time on task, accuracy toward goal completion, and other metrics.

Guiding Perspectives

Rouse III (2005) notes that video game design often starts from one of three perspectives: story, technology, or gameplay. Functionally speaking, these perspectives are not fully distinct from one another. As Schell (2005) points out, for instance, stories and games have many similarities, such as being experiences in which participants are mentally active and undergoing linear experiences. Despite their similarities and overlap, we can still consider these elements distinctly, at least for the purposes of idea generation. Rouse III (2005) describes the importance of a synergistic relationship between these three areas in order to sustain a compelling and enjoyable user experience within the game. If the technology is sophisticated and awe-inspiring but the gameplay is terrible, often the player will not play the game long enough to appreciate the technology. Awful stories will similarly dissuade many types of players, as will game engines

that are filled with glitches and based on unreliable technology. For our game, we wanted to consider each of these three areas in relation to our overall design goals, then choose a primary area and let our design emerge from there. Based on our desire to observe the impact of design changes on player experience, we chose gameplay as a starting point. However, we also considered both technology and narrative in relation to those gameplay goals. Each of these three areas is discussed briefly in this section.

Narrative Ideas

For our game's story, we decided to begin with a somewhat clichéd idea, trans-dimensionality, and then allow our game design to drive the evolution of the narrative. It is the player's story that is often more memorable than the back story devised by developers (DeMarle, 2007; Rouse III, 2005), so we wanted to employ strategies that would help blend the player's own experiences playing the level with environmental assets and dialog that would support our theme. This somewhat unusual approach meant that the narrative, like the level design and game mechanics, was iterative. Our initial story is this: due to a lab accident, a maintenance employee has been transported into an alternate universe through another dimension. The player must use her abilities to find her way back to the original universe, all while dealing with multiple other universes, each with their own challenges, along the way home. To make matters worse, the player has also been transported into another form of being – in this case, a high tech Corgi equipped with a jetpack and a punchy attitude. Although the primary reason for our selection of a Corgi for the player avatar was due to the availability of this art asset in a package from the Unity Asset Store, a web-based store where developers can acquire already made game assets (e.g., graphics, code snippets, and so on), we also liked the idea of using an animal as a unisex representation of the player. We hoped it would create a better connection between the game and the human that played it, regardless of the player's real world, biological sex.

In designing our test level, we needed enough flexibility to experiment with multiple types of platform mechanics such as jumping through moving platforms in the air, dealing with multiple types of surfaces (e.g., slippery and wet versus dry and grassy), and using items such as jetpacks or potions to alter the player's size or abilities so that we could lock off certain areas of the level until other tasks were completed. We therefore divided our test level into four primary layers, which for development purposes we referred to as the stratosphere, the base level, the basement, and the sub-basement. To support this concept narratively, the initial idea would be that upon the player's first encounter with the "accident machine" in the lab, the device would first experience a malfunction and transport the player to a base level in an alternate universe. The player would then need to find her way back to the machine again to try and reconfigure the settings, but a number of obstacles within the level would need to be overcome in order for the path to become open.

Operationally speaking, this narrative strategy meant that we could start with this initial clichéd idea in order to quickly build a level without getting too caught up in distractions. We would later have the ability during user testing to evaluate its reaction with players and hone it over time. Given our ability to use custom scripting within Unity, the game engine used for development, we could implement scripts to gather information such as how many enemies were defeated vs. left alone, how much time was spent in-game vs. time spent by other players, and how many optional tasks were completed in pursuit of a broader understanding of the world and environment. Each of these in-game user behaviors provides support for the impact of the game narrative on the player's experience.

Technology Ideas

One major factor influencing our technology selection was our need to track player metrics. While modifying, or "modding" an existing game platform would provide the absolute quickest route to having a working level, commercial off-the-shelf video games are not as flexible as custom solutions. For example, they might limit our ability to write custom scripts to track player performance behind the scenes. For this reason, we chose to develop with an engine, Unity, that allowed us to write our own tracking software and configure it exactly the way we wanted it within the level. We still needed to develop quickly, however, which led us to make heavy use of the Unity Asset Store. This service allows game developers to download preexisting assets from an online archive and then use them inside their games. Instead of modifying an entire game, though, we downloaded assets we needed and wrote our own code for the other portions of the level. Now, we no longer needed to develop graphics, record sounds or music, or implement functionalities such as dialogue systems and player physics. This greatly reduced the necessary development time, an important consideration for small teams.

Unity provides a comprehensive toolset for designing game levels. Levels, referred to as scenes in Unity, can be populated with prefabs (prefabricated items), which can then be coded to respond in different ways to interactive moments within the game. Such prefabs may be distinct from assets. Assets are created external to Unity (e.g., graphics may be designed using a graphic design program such as Adobe Illustrator or Adobe Photoshop). Prefabs are created within Unity. Any game object or collection of game objects and scripts may be turned into a prefab. For example, the developer may create an enemy agent that includes the enemy's graphic, a collider, and a script that controls movement. Instead of adding all of these elements, one by one, every time that enemy makes a repeat appearance in a level, a prefab may be created from the collection of elements. Now, the developer only has to drag and drop the prefab into the level to create a fully functioning enemy. A prefab is a type of asset, but an asset is not necessarily a prefab.

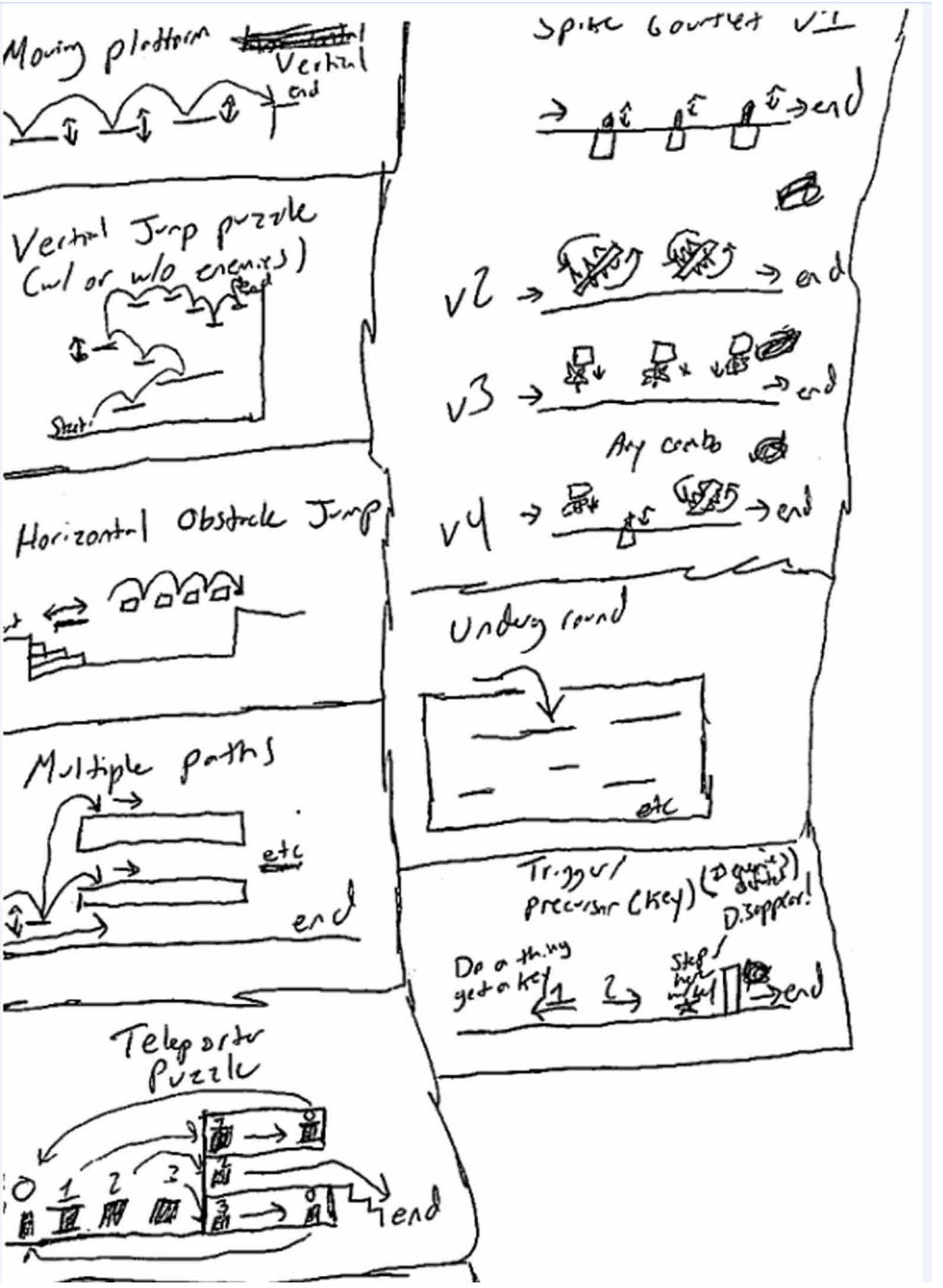
Using the Asset Store, we identified a series of prefabs and scripts relevant to our design goals and then purchased those items to give us a leg up on development work. In particular, two asset packages provided us with a base character controller model and a series of environmental objects and background effects that made it very easy to drag and drop items into our level. The objects chosen were fairly affordable, costing less than $100 to purchase. There were a number of free items available within the Asset Store, but due to our unique tracking needs, we found some well-designed assets with a minimal cost were worth the investment.

Gameplay Ideas

For our gameplay, we desired both traditional platformer mechanics (e.g., running, jumping, sliding, etc.) as well as puzzle mechanics (e.g., lock and key scenarios to limit player progression to specific areas of the level until certain items are acquired) in order to make our gameplay both familiar and intriguing to players. In terms of project management, then, one of our first steps was to sketch "rhythm groups" (Smith, Cha, & Whitehead, 2008) that defined the types of interactions we wanted to see in our level. We used these rhythm groups as a basis for subsequent development within Google Drawings, which is discussed in the next section. The best rhythms would then later evolve into the core mechanics used in the game. Our initial rhythm group brainstorming produced images such as the one shown in Figure 2.

Rhythm groups, which we can conceptualize as groups of non-overlapping action within a level, are useful for thinking about the unique clusters of gameplay that define what a player does within a game. Since our approach was experimental and incremental, we purposefully devised

Figure 2. Rhythm groups for gameplay



a large set of rhythm groups in order to test various rhythm groups and identify the patterns that seemed most critical to establish the flow of our level. As Figure 2 reveals, our initial rhythm groups included a variety of gameplay categories such as navigation (e.g., progressing through a series of moving platforms), decision-making (e.g., choosing from a series of alternate pathways), and puzzling (e.g., determining the correct sequence of teleportation to pursue in order to end up in the right area of the level to obtain a necessary item for progression). Central here is that there was much variance in between the beginning of the level and its final objective – what Gunter, Kenny, and Vick (2006) refer to as its "critical path" (p. 13) – we wanted the player to have a lot of flexibility in between the level's point of entry and its goal state. Rouse III (2005) notes this type of nonlinear design as a key aspect of what players desire in their games.

From Concept to Execution: Planning the Project

Once these core ideas about narrative, technology, and gameplay were established, we then transitioned into our design and development phase. Here, we moved our planning tools from pen and paper to more flexible, digital software tools. In the next section, we discuss how we advanced this project from the conceptual stages into some semi-formal planning processes that helped us to quickly prototype a playable level.

PHASE 2: DESIGN AND DEVELOPMENT

Guiding Themes and Challenges

Level design involves the connection of a diverse array of components into a functional product. Such a task is filled with possibilities to introduce flaws in the level, where one seemingly minor design change introduces a daisy chain of problems elsewhere in the level. Key to this process is the linkage between preproduction and production activities. A large number of developmental failures in video games seem to be traceable to the transition between preproduction and production phases (Callele, Neufeld, & Schneider, 2005). This suggests preproduction is an important stage of a video game project's life cycle, playing a major role in reducing the frequency of unforeseen issues in production (Bethke, 2003). Thus, in our own accelerated preproduction schedule, we needed to create a detailed design that would also serve as a tool to facilitate the level's development.

Our preproduction process was guided by a number of factors. Typical to other types of games, reducing technical bugs and errors was obviously one primary concern. However, our game also had some specialized concerns which guided preproduction. In this case, enabling knowledge sharing and transfer was a critical design goal. Since this was to be a game with embedded experimental manipulations, the game needed to be a vehicle for the exchange of information between developers and players. In a typical game, this mainly flows from developers to players, as players act on the information that was already set in place (i.e., through continuous interactions with the game). Players must understand the game's controls, narrative, allowable actions, enemy types and abilities, puzzles and methods for solving, and so on. However, it is rare that the developers require information from the players, except for the purposes of testing and evaluation. Communication in a game for experimentation, on the other hand, is more bidirectional between players and developers. Not only must the player respond to design cues, but the way the player interacts with the system must make it back to the development team, in the form of data, for use by the researchers.

Our level design process therefore paid particular attention to the knowledge exchange processes between designers. Paying particular attention to knowledge exchange processes between designers is important, given what Microsoft researcher and HCI designer Bill Buxton describes as the two myths of the design industry: "1) that we know what we want at the start of a project, and 2) that we know enough to start building it" (Buxton, 2007, p. 77). Typically game projects accomplish this knowledge exchange through the development and sharing of a common game design document, a document that thoroughly and explicitly details the design of the game (Callele, Neufeld, & Schneider, 2005). We have already addressed some of the challenges with GDDs above; for our purposes, we needed something more flexible to outline our design requirements in an organic and emergent way. Following Buxton's ideas, we acknowledged that sketching out details would be important for identifying gaps in knowledge that we needed to fill in through design decisions. This process is not completely novel, and has been used by at least one other small game development team with a limited budget (Kelly et al., 2007). It is one method for defining the scope of a project and working through one of the most common problems in user experience design, which is a general reluctance or even "an unwillingness to define requirements" (Garrett, 2011, p. 58).

Planning for Knowledge Exchange

To facilitate near real-time knowledge exchange during preproduction, a collaborative synchronous drawing tool, Google Drawings, was used as the primary level design and communication tool. Google Drawings is a free to use web-based software application. It allows for freehand drawing, typing, and the use of premade and custom symbols/shapes. Perhaps most importantly, it allows for multiple people to work on the same image at the same time, adding to-, deleting, or modifying participants' work in real-time. It also includes a text-based chat system that artists can use for communication while working. While not explicitly designed for the purpose of level design, it seemed like an interesting tool that could be useful for giving multiple designers an understanding of the current state of the level's design and to provide input on that design as needed.

Although the tool was helpful, several challenges were encountered. While Google Drawings provides a fairly large set of shapes and symbols, the software was not developed for the purpose of level design. As a result, these shapes and symbols were often less than ideal for our purposes. It was easy enough to use sequences of lines of rectangles to represent the ground and platforms, but other objects were more difficult to represent. For example, the designed level uses locks and keys for one type of gameplay barrier, yet none of the included shapes resemble these items. As synchronous drawing ensued, and one designer placed a hexagon with the text "K-1," the other designer became perplexed about the meaning of this abstract icon. Google Drawings does support the importing of external images, but such a process becomes tedious and time-consuming, making for inefficient level design. An alternative solution was needed. A quick discussion cleared up the confusion, and spurred the subsequent development of a key (Figure 3) to denote the meaning of the document's symbols, facilitating the development of a common mental model for level component representations.

Soon thereafter, a similar issue arose. While the key enabled a common understanding of the symbols, some symbols needed to be reused for different purposes. For example, a rectangle is well-suited for use as a platform that players should jump onto, a small wall that players should jump over, or a barrier that players need to pass. Suddenly, the exact nature of each rectangle was becoming difficult to decipher. Color-coding was the solution – using a light orange color, elements could be designated as barriers. Originally created for use in designating rectangular

Figure 3. Key of level design symbols in Google Drawings

barriers, this color quickly became useful for other types of barriers, such as a series of dangerous obstacles (e.g., time trial challenge in which the player needs to make it past several lowering gates), or places where the player needs a particular item to pass (e.g., keys, speed boost boots, or shrinking potions). Text was used to further specify each obstacle. While a series of triangles was used to designated spikes, the exact size and space for clearance were difficult to denote in Google Drawings, leading to an incomplete representation of the level portion, and, consequentially, a discrepancy between our understandings of these objects. When the triangles were colored orange (i.e., designated as a barrier), and enhanced with text (e.g., "Need shrink potion to make it past"), a shared understanding could be achieved, increasing the effectiveness of the Google Drawings as a level design tool.

While shared understandings of the tool itself were important, a shared understanding of the rules for using the tool was also important. In this project, we as designers understood that our drawings / representations / ideas might be modified in order to better accommodate the overall purpose of the level. Keeping this in mind, from the beginning, not only helped us to avoid conflict, but also helped us keep an open mind and take risks with our design decisions. In this way, design ideas were more open to elaboration than full replacement – our focus was on gradual improvement, not immediate perfection. Such a situation may not work as well with designers who are not familiar with — and respectful of — each other's work. For that reason, this process works better for small groups of designers who have established trust and are already well-acquainted with one another.

Planning for Risk and Uncertainty

Coding errors produce a game that functions poorly or behaves unexpectedly. While problematic in all circumstances, it becomes particularly problematic when developing a game for controlled experimentation. Games already provide an environment which contains a lower level of control than is comfortable for many researchers. The very term "game" implies that users will be able to exert some degree of choice (Malone & Lepper, 1987). Giving players the freedom of choice necessitates a certain degree of unpredictability – a lack of control. Thus, a fully controlled environment does not accurately replicate normal gameplay, making it a necessary evil. While games in experimentation should not be fully controlled, the undesirable variety of variability is still a concern. Bugs and errors almost always fall into this category.

To increase our understanding of potential bugs and errors within the context of our entire project, we developed a risk breakdown structure (RBS), a document that is familiar to most project managers, both in and outside of game projects. The RBS included a closeout plan column for addressing various types of complications within the level (Table 1). A RBS categorizes the risk factors by group or category (Holzmann, 2011). While a closeout plan typically ensures that all project requirements have been met (De Furia, 2008), our closeout plan paid particular

Table 1. A portion of the risk breakdown structure

Risks – Level 1	Level 2	Probability	Impact	Mitigation Strategy	Closeout Strategy
Gameplay	Play time too short to observe variability b/w participants	Moderate	High – Unable to detect differences / Type 2 error	Consider length of time between each major challenge, and length of time to complete each challenge	Playtest – Speed run to identify quickest possible completion time
	Game not fun. Participants quit playing quickly	Moderate	High – Unable to detect differences / Type 2 error	Make sure Gameplay is varied. Avoid Cliches. Be innovative	Playtesting with people who do not hold stake in the game's development / success
	Game too fun. Participants do not stop playing	Low	Moderate – Unable to detect differences / Type 2 error	Identify a maximum play time. Hold particpants to that maximum	Playtesting with people who do not hold stake in the games game's development / success
Technical	Bugs introduce unwanted variation in gameplay b/w participants	Moderate	High – can make data unreliable / Type 1 error	Frequent testing during development	Playtest – Slowly explore every aspect of the game. Anticipate player interactions. Try to break the game.

attention to the risks, ensuring they did not manifest. In this project, the combined RBS and closeout plan composes the backbone of the game's testing.

Not only does the RBS form a plan for testing the occurrence of each anticipated risk, it also serves as a written document that is accessible by all team members, creating a shared understanding of the potential problems that could arise during development. When applied more globally to larger projects, this shared understanding is critical to ensuring that all team members maintain awareness of the risks and actively work to mitigate those issues. In this manner, the risk breakdown structure performs a role at all phases of the project's development, facilitating a smooth project flow for those working on its design and development.

DISCUSSION

We began this research by considering two research questions: what project management strategies are effective for small games projects, and how do these strategies provide insights regarding the larger user experience of game designers? After analyzing our data and the two phases of this case history, we have reached several conclusions.

First, in regards to the strategies that proved most effective for product management and knowledge exchange, a number of tools and techniques proved useful for us. These included:

1. An *open, informal model for regular meetings*. During the idea generation phase and on into development, we found it very valuable to exchange ideas informally, when inspiration seized us, rather than relying upon standing formal meetings at particular times. This open and accessible model for information exchange is also apparent in larger independent studios (McDaniel, 2015), but it also worked well for the small paired design task we undertook here. Moreover, it draws similarities to Scrum methodology, which prescribes quick daily meetings to bring team members up to speed and discuss current challenges (Keith, 2010). While Scrum meetings are once a day, our meetings were on an "as you need it" basis for exchanging information. In practice, meetings occurred several times per day, and rarely exceeded 5 minutes.
2. A *willingness to use community-authored assets* to speed up production during prototyping. We made heavy use of the Unity Asset Store, but there are many other community-oriented digital media sites that concentrate on specific digital assets (e.g., sounds, animations, video clips, and 3d character models). For small teams of developers, such resources are helpful, particularly when one's design talent does not encompass every possible variation of game design. Although cost varies depending on the type of asset used, many assets are quite affordable.
3. Adopting *visual sketching tools* early on in the process. Our user experience sketches began with the primitive "back of napkin"-type sketches shown in Figure 2, then matured into collaborative, digital versions using Google Drawings. Google Drawings made it very easy to quickly see key design challenges, as we discussed in Phase Two, and articulate obstacles that we had not immediately seen before converting our ideas into a visual form. Before selecting Google Drawings as our tool of choice, we researched a number of other alternatives before deciding Google Drawings included the best balance between usability and features.
4. Building *risk minimization strategies* into our design and development pipeline that allowed us to make good decisions early in the process. While the risk breakdown spreadsheet shown in Figure 4 may seem needlessly complicated for designing a single level, our ambitious goals to build this level as an experimental platform for research made it useful as a focusing tool for the early design decisions that would continue to influence the work as it grew in complexity.

In regards to our second research question, considering what we can learn about the user experience of developers based on this reflexive analysis, there are a number of insights we gleaned from this process:

1. **Genre is Exceptionally Important to the User Experience of Game Design:** Whereas our experiences were focused on rhythm groups of action and building pathways from one point to another within our level, these were mechanics suitable for the action/platformer genre. In contrast, a role playing design experience requires its builders to be much more involved with the narrative world surrounding gameplay. Our story was not given much weight as we wanted the gameplay to help drive the story, rather than the other way around. Similarly, race car simulator games and real time strategy games will have their own specialized factors which must be considered during design. Available assets may also influence choice

of genre. Since we needed to use *community-authored assets* to speed up development, we were constrained by what was available – partially in terms of graphics, but especially in terms of available scripts.

2. **Rapid Iteration During Design and Development Draws Similarities to Player Experimentation in Gameplay:** Just as players will rapidly experiment with various gameplay tactics in order to determine optimal strategies for success, so can developers quickly iterate through different design scenarios to find a good fit for their audience. In our case, *adopting visual sketching tools* and *regular informal meetings* facilitated this rapid iteration. We found that once we had some concrete ideas mocked up in Google Drawings, it was easier to see how to move elements around in order to combine moments of frenetic activity with the necessary recovery time to allow players to gain back their bearings. Iterative design became a strategy that was applied not only with early builds of the level, but also with each revision to the level design process. As discussed in Phase Two, we also used iterative design tactics to work out solutions to our design challenges, such as the lack of a robust stencil set for shapes and the lack of a common icon system for designating different elements within the level.
3. **Playtesting Remains the Gold Standard of Risk Mitigation in Game Development:** As we considered each of the risks we identified in our risk breakdown structure (Figure 4) we realized that we could not truly comment on any of these areas until we had acquired data from actual player experiences with our system. As developers, then, we could spend a great deal of time in the project management phase *building strategies for risk minimization*, but without actually seeing how players interacted with our level, we had no way of accurately classifying these risks in terms of probability or impact, although we did make informed guesses based on our prior experience developing similar systems. Playtesting is critical for devising strategies for dealing with these types of risks as the project matures.

LIMITATIONS

This article focuses on a unique type of game designed with experimental manipulations as a key feature. These project management strategies must be evaluated in other settings, with other small design teams, and other games, to identify their robustness to a variety of alternate conditions and contexts. We also had particular resource limitations. Namely, our need to rapidly develop with a limited budget meant that we needed to use pre-made assets. These assets necessarily influenced the design of the game. While our small team did not include a dedicated graphic designer, other small teams could have someone fulfilling that role, making visual asset development a more efficient pursuit, and possibly changing the dynamics, requirements, and limitations of the team. Taking these factors into consideration in future studies will extend the present research.

CONCLUSION

In this article, we considered the design and development process of a small experimental game level with a specific purpose in mind: configuring level elements in different ways to measure the impact of design on player behavior. While the design process is atypical of large commercial games, in that there were only two designers who worked to build a single level over the span of several weeks, it is still important to understand these topics for a number of reasons. First, even large commercial games are often decomposed into smaller tasks that a small number of

individuals will need to work together to complete. This means that the knowledge management tactics and design strategies developed in the pursuit of small goals are still relevant to larger projects. Second, the tactics studied here, such as the reliance on preexisting community assets (e.g., Unity Asset Store), sharable online design tools (e.g., Google Drawings), and project management techniques (e.g., risk breakdown structures) are also each helpful for more complex projects, so it is useful for us to understand how these types of tools are used in real world scenarios. For example, a large team working for a commercial games studio might want to prototype a new model for gameplay, but finds that designing entirely new art assets is cost prohibitive. In this case, a community asset store saves a tremendous amount of time and money. Similarly, a producer might want to weigh decisions within the context of risk as crunch time begins, meaning that a risk breakdown chart holds significant value for this person.

Future research should continue to expand this line of inquiry in a number of directions. In addition to small, two person teams, it is important for us to better understand the knowledge exchange and project management practices of mid- to large- size game development teams. Similarly, we need to better understand the relationship between project management and user experience in other domains outside of video games. Finally, within the area of games, it would be interesting to see if the same strategies and practices that work well for the design and development of platformer video games also apply to other types of genres. In most cases, different project management skills are likely to be important. For example, as previously mentioned, role playing games have a much heavier emphasis on story development and character dialog. Within this genre, other techniques such as storyboarding and live action role playing might play a useful role alongside collaborative drawing tools and risk breakdown sheets.

REFERENCES

Allen, J. P., & Kim, J. (2003). Digital gaming evolution: A sociotechnical approach. *Proceedings of the Abstracts of the Organizations and Society in Information Systems Workshop*, Seattle, WA.

Anderson, L. (2006). Analytic autoethnography. *Journal of Contemporary Ethnography*, *35*(4), 373–395. doi:10.1177/0891241605280449

Bethke (2003). *Game development and production*. Plano, TX: Wordware Publishing, Inc.

Buxton, B. (2007). *Sketching user experiences: getting the design right and the right design*. San Francisco: Morgan Kaufmann.

Callele, D., Neufeld, E., & Schneider, K. (2005). Requirements engineering and the creative process in the video game industry. *Proceedings of the 13th IEEE International Conference on Requirements Engineering*, Paris, France (pp. 240-250). IEEE Press. doi:10.1109/RE.2005.58

Chandler, H. W. (2010). *The game production handbook* (2nd ed.). Sudbury, MA: Jones and Bartlett Publishers.

Chua, A. Y. (2005). The design and implementation of a simulation game for teaching knowledge management. *Journal of the American Society for Information Science and Technology*, *56*(11), 1207–1216. doi:10.1002/asi.20224

De Furia, G. L. (2008). *Project management recipes for success*. Boca Raton, FL: Auerbach Publications. doi:10.1201/9781420078251

Dreyfuss, H. (2012). *Designing for people* (4th ed., Kindle). New York: Allsworth Press.

Ellis, C. S., & Bochner, A. (2000). Autoethnography, personal narrative, reflexivity: Researcher as subject. In N. Denizen & Y. Lincoln (Eds.), *The handbook of qualitative research* (pp. 733–768). Los Angeles: Sage.

Fanfarelli, J. R. (2014). *The effects of narrative and achievements on learning in a 2D platformer video game* [Doctoral dissertation].

Fullerton, T. (2008). *Game design workshop: A playcentric approach to creating innovative games*. Amsterdam: Elsevier.

Garrett, J. J. (2011). *The elements of user experience: User-centered design for the web and beyond* (2nd ed.). Berkeley, CA: New Riders.

Greene, J., & Palmer, L. (2014). It's all fun and games until someone pulls out a manual: Finding a role for technical communicators in the game industry. In J. deWinter & R. M. Moeller (Eds.), *Computer games and technical communication: Critical methods and applications at the intersection* (pp. 17–33). Burlington, VT: Ashgate Publishing.

Gunter, G. A., Kenny, R. F., & Vick, E. H. (2006). A case for a formal design paradigm for serious games. *The Journal of the International Digital Media and Arts Association*, *3*(1), 93–105.

Hight, J., & Novak, J. (2007). *Game development essentials: Game project management*. Clifton Park, NY: Thomson Delmar Learning.

Holzmann, V., & Spiegler, I. (2011). Developing risk breakdown structures for information technology organizations. *International Journal of Project Management*, *29*(5), 537–546. doi:10.1016/j.ijproman.2010.05.002

Hunicke, R., LeBlanc, M., & Zubek, R. (2004, July). MDA: A formal approach to game design and game research. *Proceedings of the AAAI Workshop on Challenges in Game AI* (Vol. 4). San Jose, CA: AAAI Press.

Keith, C. (2010). *Agile game development with Scrum*. Upper Saddle River, NJ: Addison-Wesley.

Kelly, H., Howell, K., Glinert, E., Holding, L., Swain, C., Burrowbridge, A., & Roper, M. (2007). How to build serious games. *Communications of the ACM – Creating a Science of Games, 50*(7), 44-49.

Luoto, T., Korpelainen, R., Röning, J., Ahola, R., Enwald, H., Hirvonen, N., & Heikkinen, H. I. et al. (2014). Gamified persuasion: User experiences of online activation service. *International Journal of Sociotechnology and Knowledge Development*, *6*(4), 1–17. doi:10.4018/ijskd.2014100101

Malone, T. W., & Lepper, M. R. (1987). Making learning fun: A taxonomy of intrinsic motivations for learning. In R.E. Snow & M.J. Farr (Eds.), Aptitude, learning, and instruction volume 3: Cognitive and affective process analyses. Hillsdale, NJ: Lawrence Erlbaum Associates.

McDaniel, R. (2015). Communication and knowledge management strategies in video game design and development: A case study highlighting key organizational narratives. *Proceedings of the 2015 IEEE International Professional Communication Conference* (pp. 1-16). IEEE Press. doi:10.1109/IPCC.2015.7235773

Miller, P. (2008). Top 10 pitfalls using scrum methodology for video game development. *Gamasutra*. Retrieved from http://www.gamasutra.com/view/feature/3724/top_10_pitfalls_using_scrum_.php

Norman, D. A. (2013). *The design of everyday things: Revised and expanded edition*. New York: Basic books.

Rouse, R. III. (2005). *Game design theory and practice* (2nd ed.). Plano, TX: Wordware Publishing.

Salen, K., & Zimmerman, E. (2004). *Rules of play: Game design fundamentals*. Cambridge, MA: MIT press.

Sansone, A. (2014). Game design documents: Changing production models, changing demands. In J. deWinter & R. M. Moeller (Eds.), *Computer games and technical communication: Critical methods and applications at the intersection* (pp. 109–123). Burlington, VT: Ashgate Publishing.

Sawyer, S. (2013). Sociotechnical approaches to the study of information systems. In A. Tucker & H. Topi (Eds.), *CRC handbook of computing*. Chapman and Hall.

Schell, J. (2005). Understanding entertainment: Story and gameplay are one. *Computers in Entertainment*, *3*(1), 6–6. doi:10.1145/1057270.1057284

Smith, G., Cha, M., & Whitehead, J. (2008). A framework for analysis of 2D platformer levels. Proceedings of Sandbox '08, Los Angeles, CA, USA.

Spaulding, I. I. S. (2009). *Team leadership in the game industry*. Boston, MA: Course Technology Press.

Tuffley, D. (2009). Mind the gap. *International Journal of Sociotechnology and Knowledge Development*, *1*(1), 58–69. doi:10.4018/jskd.2009010103

Wolz, U., & Pulimood, S. M. (2007, March). An integrated approach to project management through classic CS III and video game development. *SIGCSE Bulletin*, *39*(1), 322–326. doi:10.1145/1227504.1227422

Yin, R. K. (2014). *Case study research: Design and methods* (5th ed.). Thousand Oaks, CA: Sage publications.