A Taxonomy of Design Quality in Super Mario Maker 2

Carlo A. Furia
Software Institute – USI Università della Svizzera italiana
Lugano, Switzerland
bugcounting.net

Andrea Mocci
Software Institute – USI Università della Svizzera italiana
Lugano, Switzerland
andrea.mocci@usi.ch

***Abstract*—Thanks to its popularity, Super Mario Maker 2 (SMM2) features myriad user-designed platforming levels that span a broad range of styles, genres, difficulty, and quality. This paper is a first attempt at charting out this vast landscape of levels, with the goal of exploring the main concepts that characterize *design quality*. To this end, we first selected 500 videos by popular YouTube streamers that play SMM2 levels. With the help of an LLM, we extracted and clustered by topic segments where the streamer comments about the design of the level they are playing. Then, we applied a grounded theory coding process to extract the concepts emerging in these segments, and we combined and organized them hierarchically. The resulting taxonomy identifies the main abstract concepts that pertain to the design of a level in SMM2 and its resulting quality, as well as several concrete elements of the game that belong to each abstract category. Besides illustrating the underlying dimensions that characterize design quality in SMM2, our results also illuminate the diverse landscape of player-designed levels in the game.**

***Index Terms*—design quality, Super Mario Maker 2, taxonomy, platformer, grounded theory**

I. Introduction

Super Mario Maker 2 (SMM2) is a popular[1] Nintendo Switch platformer that is built around a level editor and online features. SMM2 players can design their own levels, using many of the features and styles of classic Nintendo platformers such as Super Mario Bros. 1 and Super Mario World, and then share them publicly by uploading them to Nintendo's servers. This makes SMM2 an open-ended game with virtually infinite community-built content: at the time of writing, it features over 29 million levels.[2] Together with its predecessor SMM1 for the Wii U, SMM2 has provided an official outlet to the amateur community devoted to ROM hacking and Kaizo-style levels,[3] while simultaneously making platformer level design a widely accessible activity. As a result, SMM2 features a treasure trove of all sorts of levels, spanning a very broad range of styles, quality, difficulty and genres, and showcasing a lively, diverse, dedicated community of passionate players and designers [1]. This makes SMM2 levels, and the community that produced them, an appealing object of study to understand good and bad practices in 2D-platformer design.

Despite this potential, there is comparatively little systematic research about SMM2, especially in relation to design quality. The present paper is a first attempt to fill this knowledge gap: our overarching goal is charting out what goes into the design of a level in SMM2, and especially what dimensions characterize a level's *design quality*. Concretely, we build a *taxonomy* that summarizes the level design process and relates concrete elements (such as an enemy placement) to abstract concepts (such as a level's genre or difficulty).

To this end, we tapped into the knowledge and experience of 5 popular YouTube content creators whose production focuses on SMM2. Precisely, we selected a sample of 500 of their YouTube videos, where they frequently engage in explicit commentary about the quality of the SMM2 levels they were playing, and about the features that underlie those levels' design; these commentaries provided the basic data for our taxonomy building. To sift through such a massive amount of data, we extracted, with the help of an LLM, relevant video segments where the streamer argues about design quality, and clustered them by similar topics. Then, we applied methods from grounded theory [2] (in particular, various forms of *coding*) to analyze the selected segments and extract keywords and concepts that characterize a level's makeup. Finally, we organized these concepts into a hierarchical *taxonomy* that articulates the different dimensions of level design quality.

The taxonomy we built is useful to better understand the ingredients and dimensions underlying the design of a platformer level, and to showcase the distinct design culture and practices of the SMM2 community. It is also a first step on the way to a rigorous and systematic understanding of what characterizes the design quality of a platforming level, which, we hope, can support further work in this direction.

Data availability. For reproducibility, the details of the taxonomy construction process are available in a replication package.[4]

II. Related Work

General game and level design. While there are plenty of resources on game design, Smith et al. [3] note that game

[1]According to Wikipedia, it is the Nintendo Switch's 24th all-time best-selling game.

[2]https://smm2.wizul.us/

[3]https://www.smwcentral.net/

979-8-3315-9476-3/26/$31.00 ©2026 IEEE

[4]https://dx.doi.org/10.5281/zenodo.20423865

design books usually present general principles rather than focusing specifically on *level* design. In the few exceptions that focus on level design, the presentation of concepts and principles is usually also generic rather than *genre*-specific. For example, Feil and Scattergood [4] only include a short section on level design guidelines that are genre-specific. Totten [5] focuses on specific aspects of level design (e.g., spacial layout, aesthetics, and rhythm) with examples from a few genres (e.g., first-person shooters and RPGs). Salmond [6] covers principles of good level design from a deliberately genre-agnostic perspective. Canossa and Smith [7] introduce metrics to evaluate the quality of level design in general, and demonstrate them on *Super Mario Bros.* and *Portal 2* levels. Other approaches use qualitative methods to present patterns for specific genres, such as first-person shooters [8] and RPGs [9]. In all, most of the literature on level design is not specific to platformers, and focuses on "mainstream" design principles that may fail to cover the diversity of user-designed levels present in a game like SMM2.

Platformer level design. Smith et al. [3] is one of the earliest design analyses that focuses on platformers—specifically, on the *items* used in platformer levels and how they are used to construct "areas of challenge". Based on their experience and observations, Khalifa et al. [10] discuss level design patterns such as *safe zones*, *branching*, and *pace breaking*, which are applicable to a range of 2D games (including, in particular, platformers). Through a manual annotation process of levels in 10 different games of the *Super Mario Bros.* series, Thompson [11] extends the patterns identified by Dahlskog and Togelius [12] and adapts them to the specific features of the popular Nintendo series. Each pattern usually characterizes a certain combination of game items and is assigned to a category (e.g., *path* or *enemy*). Summerville et al. [13] introduce layout metrics, such as "enemy density", and propose to use them as predictors of difficulty, visual aesthetics, and enjoyment as perceived by players. Eloe and Meyer [14] provide an interesting analysis of *indicators*: bespoke visual elements used to guide the player in SMM2 levels.

Taxonomies in game research. Taxonomies have been used extensively in (video-)game research to characterize aspects such as bugs [15], failure and challenge for the player [16], [17], and mechanics like respawn [18]. To our knowledge, Cuerdo et al.'s taxonomy of death and rebirth [19] is the only one that focuses on platformers.

User-generated content (UGC) has become an increasingly prominent aspect of the video game industry [20]. The recent survey by Duan et al. [21] provides a framework to characterize different forms of UGC according to various dimensions such as game experience. The survey also identifies open research questions that motivate further user-focused work; among them, "How do players conceptualize quality in user-created levels?" is relevant for the present work.

Summing up. Level game design has been extensively analyzed in general, but few studies focus on level design for *platformers*. Another, increasingly relevant angle that is underrepresented in research is design quality for *user-created* levels, which may feature characteristics and variety different from professionally-produced ones. Our work tries to bridge these gaps by focusing on the popular platformer game SMM2, whose content is almost entirely created by regular users.

III. Methods

This section describes the process we followed to address the paper's main research question:

What concepts characterize the design quality *of a level in* SMM2*?*

and to build a taxonomy that answers the question. The methods described here are primarily *qualitative*, which are appropriate when exploring an area not yet thoroughly researched [2].

A. Data

Our approach starts from the following observation, based on the authors' personal experience watching SMM2 videos on YouTube: popular streamers usually do not just play levels for the fun of viewers, but simultaneously engage in verbal commentary that articulates their reactions to, and opinion of, the levels' design quality. While the style, verbosity, frequency, and tone of comments varies with each streamer's personality, even the quietest streamer routinely expresses their frustration or delight with the level they're playing. Especially for streamers with an extensive experience with SMM2 and similar platformers, such commentary often includes remarks on the levels' design from different points of view, ranging from technicalities specific to SMM2 (e.g., the hitbox of certain obstacles) to high-level observations (e.g., what defines a certain genre such as troll levels). Thus, given this paper's goal, the videos produced by experienced streamers provide a high quality data source to base a taxonomy of concepts related to design quality in SMM2 levels.

Streamer selection. We selected the five streamers with the following YouTube handles: `GrandPOOBear`, `PangaTAS`, `DGR_Dave`, `LilKirbs`, and `Thabeast721`. They are among the most active YouTube creators focusing on SMM2 content; the first two have also been active as makers (level creators); another reason for selecting them is that we (the authors) are familiar with their videos, and appreciate their expert knowledge of the game. All five streamers produce content in English—which is also the most used language in SMM2 levels [1].

Even though some of the selected streamers are also active on other platforms, we specifically focused on their YouTube playlists that contain *edited*[5] videos where they play popular challenges in SMM2, such as *Super Expert No Skip* and *uncleared levels*. The former is a challenge where a streamer tackles SMM2's Super Expert Endless Challenge—trying to beat as many randomly selected "super expert" levels as possible before running out of lives—with the additional, self-imposed

[5]Edited videos are more content-dense than unedited live streams, and typically feature the streamer's comments without frequent interruptions from interacting with the viewers' live chat.

constrain that no level can be skipped, no matter how difficult or frustrating it is. The *uncleared levels* challenge simply tackles any level (usually of "expert" difficulty) that has not been yet cleared by any players. Such challenges provide a good variety of level content and quality, and thus showcase the streamers' wildly different reactions.

Playlist and video selection. For 4 out of 5 streamers, we selected 2 different playlists and sampled 50 videos from each playlist. On average, each video consists of about 30 minutes of gameplay, usually covering 5–15 levels. `LilKirbs`'s YouTube videos have a significantly different style, as they are typically only around 2 minutes long, and each focuses on clearing a single, very challenging level with detailed commentary. Therefore, we sampled 100 videos, all from the same single `LilKirbs` playlist. When sampling from a playlist, we stratified over different years, trying to select roughly the same number of videos published in each year covered by the playlist. In total, we sampled exactly 500 videos spanning the years 2019 to 2025, consisting of over 205 hours of commented playthrough content.

Video segmentation. Given the extended amount of sampled content, watching and analyzing *all* of it would require an excessive amount of time. Furthermore, we note that the density of commentary in the videos is typically low, with a few comments relevant to design quality interspersed in a longer stretch of video—and somewhat repetitive—as numerous concepts are recurring in multiple videos. To render the coding process (Sec. III-B) feasible in a reasonable amount of time, we used an LLM (Large Language Model) to identify the relevant segments of videos.

To this end, we extracted the transcript of each video using the `yt-dlp`[6] command line tool. The tool encodes transcripts using the WebVTT format, a text-based format for HTML5 video subtitles. We then designed a prompt whose purpose is to extract a set of relevant segments, where the streamer discusses aspects related to design quality. To this end, we tried a few different prompt variants on a sample video, and selected the variant that produced the clearest and most systematic segment extraction. The core part of the selected prompt[7] that instructs the LLM is as follows:

> *Extract transcript segments where the streamer is describing or commenting on aspects of level design. These aspects can be good/enjoyable, bad/fustrating, fair/unfair, and so on.*

The prompt also instructs the LLM to extract the start and end timestamps of the relevant segment, its transcript text, and a short text with a reason for inclusion. We then submitted the prompt for each video transcript to `gpt-5.2` [22] with *medium* reasoning effort and collected the results.[8] The segmentation task produced 6,752 segments; we removed 47 outliers because they were either too short (less than 1.5 seconds) or too long (over 5 minutes), leaving us with 6,705 segments.

Segment clustering. The number of extracted segments is still somewhat large for coding; precisely, we noticed a significant degree of repetition, with several segments expressing essentially the same judgment or observation about different levels. To streamline the coding process, we applied a clustering algorithm to identify segments that contain commentary with similar content. To this end, we selected one of the best performing *embedding models* for natural language clustering according to the *Massive Text Embedding Benchmark (MTEB)* [24] leaderboard:[9] `Qwen/Qwen3-Embedding-4B`,[10] which features an embedding dimension of 2,560. For the embeddings construction, we joined, for each segment, the transcript and the reasoning generated by the LLM. We then adopted a standard clustering approach on large embeddings combining UMAP [25] for dimensionality reduction and HDBSCAN [26] as proper clustering algorithm.[11] This produced 188 clusters covering 4,773 segments and 1,932 outlier segments. For each cluster, we selected one exemplar segment, which is a representative member of the cluster. An exemplar in HDBSCAN is analogous to the concept of centroid in traditional clustering algorithms like k-means.[12] The resulting 188 exemplar segments underwent the coding process described next.

B. Coding

Open coding. The first step of data analysis followed an *open coding* process [2], whose goal is identifying the concepts in the data that pertain to the research question. To this end, we implemented a simple web-based UI consisting of several pages; each page embeds one of the 188 video segments (extracted as described in Sec. III-A), its corresponding transcript, and the LLM's reasoning of why the segment is relevant to design quality. The page also allows users to type in any number of *codes* (i.e., keywords) and to select a previously inserted code (from a pull-down menu or using autocompletion). Each author, independent of the others, inspected all 188 extracted segments and annotated them with codes; this produced a total of 289 unique "raw" codes.

Axial coding. The codes collected with open coding were the input to an *axial coding* process [2], whose goal is merging the codes and relating them to the context. In a series of meetings involving all authors, we perused all codes, and combined them into a single, unified set. Concretely, the axial coding process merged codes that were judged to represent the same concept (e.g., *checkpoint placement* and *checkpoint presence*), split "compound" codes into their components (e.g., *cheese alternate route* into *cheese*—a way of progressing through a level that evades the designer's intention, often by exploiting glitches or design oversights to drastically

[6]https://github.com/yt-dlp/yt-dlp

[7]The full prompt is available in the replication package.

[8]At the time of writing, `gpt-5.2` ranks as one of the best models (rank 4 in the Nanonets-IDP benchmark [23]) for similar tasks like *Key Information Extraction*.

[9]https://huggingface.co/spaces/mteb/leaderboard

[10]https://huggingface.co/Qwen/Qwen3-Embedding-4B

[11]https://umap-learn.readthedocs.io/en/latest/clustering.html

[12]For an explanation of how exemplars work in HDBSCAN see https://hdbscan.readthedocs.io/en/latest/soft_clustering_explanation.html.

simplify or shorten a section—and *alternate route*), and dropped a few codes that were deemed not relevant to the research question (e.g., *world record*, which characterizes the fastest time taken by a player to clear a level, and is thus immaterial for the design process because it happens after design is complete). This produced 112 unique "refined" codes.

Summary memos. In the next step, we produced *memos* that summarized the main concepts encapsulated by the curated set of codes produced by axial coding. All authors participated in this *memoing* process, which took part in another series of discussion meetings; usually, one author was in charge of materially typing the memo for a certain concept, while the others contributed to the discussion and suggested changes or additions. Concretely, the memos provided definitions of the high-level, abstract concepts, and gave concrete examples of the relations between abstract and concrete concepts. For example, this process identified the fundamental distinction between a level's spatial and temporal constraints; assigned the codes *layout* to refer to the former and *timing* to the latter; and connected the concrete feature of *autoscroll* to the *timing* abstract concept.

C. Taxonomy Construction

Theoretical integration. To finally build a taxonomy that refines the information collected in the memo and organizes the categories into a hierarchy, we followed a process of theoretical integration [2] in four steps. First, we identified the central category (*design quality*), which follows naturally from the targeted research question. Second, we worked our way through the various codes and partitioned them into *abstract* and *concrete*: abstract codes label concepts that are general and subsume different concrete elements, whereas each concrete code identifies a specific element in SMM2. Third, we organized the abstract codes into a hierarchy (nodes with red background in Fig. 1), starting from the central category and proceeding top-down. Fourth, we built a grid with one row for each concrete code and one column for each leaf of the abstract hierarchy tree, and we assigned each concrete code to the abstract concept that it exemplifies.

Validation. We validated the taxonomy in two, complementary ways. First, we relied on our (the authors') experience with and knowledge of SMM2[13] to identify any obviously missing or inconsistent elements in the taxonomy, especially for what concerns concrete elements. Among other things, we clarified that the attribute *scuffed* (one of the original codes) is used with a generic meaning of "poor quality";[14] as a result, it only serves as a general characterization of design quality, rather than being a form of *jank* (a flawed, poorly designed or implemented setup) as we initially thought. Another example of taxonomy change introduced by validation is the inclusion of the concrete codes *pixel art*, *short and sweet*, and *CCBE*: these did not emerge directly from the open coding process, but we realized, during the validation phase, that they are prominent examples of concrete concepts corresponding to various abstract output categories (as we explain when we present the taxonomy in Sec. IV-A2).

The second form of validation followed the LLM-as-a-judge approach [27] using `gpt-5.4`[15] on the transcripts of the "outlier" cluster (see Sec. III-A). The objective was to assess whether the 1,932 outlier segments include information that is not covered by the taxonomy. To this end, for each outlier segment o, we prompted the LLM with two requests in the same prompt: *i*) confirm whether o's transcript and reasoning for inclusion (provided during Sec. III-A's video segmentation process) are still relevant to SMM2 level design quality; *ii*) match, if possible, o's content to an abstract concept (precisely, those in the level just before the concrete leaves, such as *item*, *setup*, etc.) in our taxonomy—which was given in its entirety in the prompt. In over 99.7% (1,927) of all 1,932 segments, the LLM successfully matched the content to taxonomy concepts. The 5 exceptions are segments that lack a prominent, clearly defined content; hence, they could match several different categories in our taxonomy. In all, the validation confirms that out taxonomy adequately covers the design quality concepts expressed in the videos.

IV. Results

A. A Design Quality Taxonomy

Fig. 1 displays the taxonomy, whose elements we describe in detail in this section. For space reasons, the figure does not include all leaves—that is, the concrete elements that belong to each abstract category. Instead, it shows one example of concrete element for each category; the following text mentions more concrete examples, whereas the replication package includes all those that emerged from the coding process (which are still not expected to be exhaustive).

At the top level, all factors that concur to the *design quality* of a SMM2 level can be classified as inputs or outputs. This classification comes from viewing a level as the product of a design *process*: *i*) the *inputs* are what the maker directly selects and combines to build a level; *ii*) the *outputs* are the results of the input in terms of achieved quality factors.

1 Inputs: are of two kinds: *ingredients* and *contraptions*.

1.1 Ingredients are the inputs to the design process of a level that are directly and concretely under the maker's control. We list them roughly in order from basic to complex.

1.1.1 *Item*: any element that may be included in a level. This comprises all sorts of assets[16] such as blocks, terrain, water, *power-ups*, pipes and doors, enemies, the goal, checkpoints, and so on.

1.1.2 *SMM2 features*: the capabilities of the SMM2 game engine, its editor, and its communications with Nintendo's

[13]Integrating the researchers' knowledge with the data's information is a standard practice in qualitative research [2, Ch.1]. For our work, we leaned mainly on the data's, but we still integrated our own knowledge for validation.

[14]https://en.wikipedia.org/wiki/Glossary_of_video_game_terms#scuffed

[15]This specific version was just released at the time of validation.

[16]We use "item" with a more general meaning than how it is used within the SMM2 editor, where "item", "gizmos", and "enemies" all denote distinct subsets of game elements.

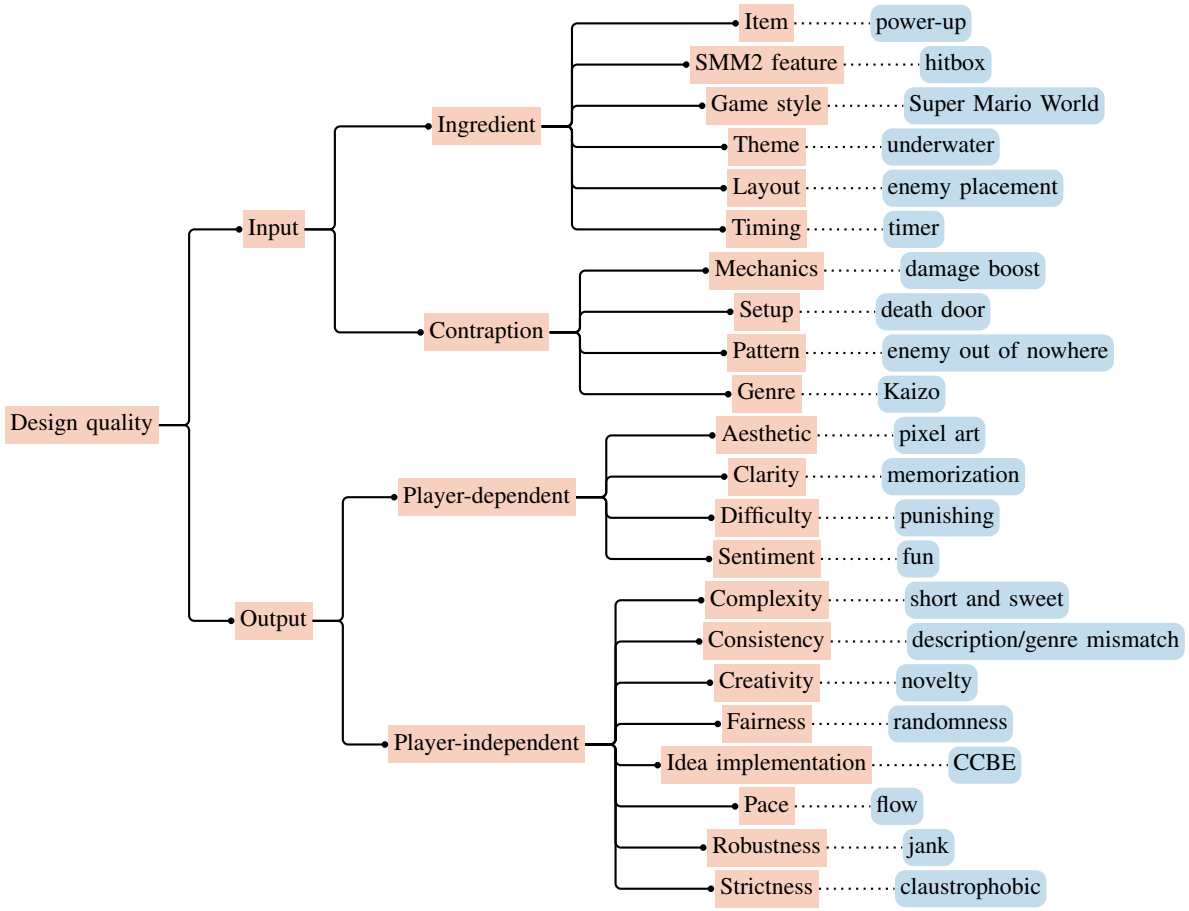

Fig. 1: A taxonomy of design quality in SMM2. Nodes with red background denote abstract categories in the taxonomy, whereas nodes with blue background denote examples of concrete elements that belong to each category.

servers. This ingredient is fixed for all SMM2 level designs, and it defines the design space for all makers. An example is the *hitbox* of enemies or obstacles, which constrains how players should approach them.

1.1.3 *Game style*: A subset of features of the game engine with a distinct graphical flavor. SMM2 offers five game styles, based on classic Nintendo games: Super Mario Bros. 1, Super Mario Bros. 3, *Super Mario World*, New Super Mario Bros. U, and Super Mario 3D World.

1.1.4 *Theme*: A particular graphical style. The theme may also constrain the availability and behavior of items in a level, as well as its physics. For example, the *underwater* theme supports swimming, whereas the snow theme uses ice physics, which renders surfaces slippery.

1.1.5 *Layout*: The spatial arrangement of items in a level. A level's layout is the central design ingredient, as it is the practical, physical realization of the maker's design ideas. The *enemy placement* is an example of a, usually important, layout component.

1.1.6 *Timing*: The direct temporal constraints of a level. The *autoscroll* (which forces the level to scroll at a certain speed), a level's *timer* (the allotted time to complete a level), or raising lava (which fills up the screen, forcing the player to reach high ground to avoid it) are all examples of timing features.

1.2 Contraptions are combinations of ingredients in a level. We list them in order from specific to general.

1.2.1 *Mechanics*: Any situation that allows for, or requires, a certain player action. For example, a power-up gives the player an additional hit, which enables the *damage boost* mechanics. Another example of simple mechanics is the *ground pound*, which is available in certain game styles if the layout gives enough room to activate it.

1.2.2 *Setup*: A specific arrangement of items that, possibly interacting with certain mechanics, introduce (intended or unintended) constraints into a level. A *death door* (a door that leads the player to certain death when entered) is an example of a simple setup. A *shell jump* is a setup usually consisting of a shell and a wall that is too tall to be cleared with a regular jump; the setup requires the player to throw the shell against the wall and jump on it as it bounces back in midair, which allows them to jump higher than in a regular jump. A setup may also involve purely visual elements, such as in the case of *indicators* [14]. A shell jump indicator marks the exact location where the player should throw a shell to shell-jump.

1.2.3 *Pattern*: A way of introducing a high-level constraint by combining setups and mechanics, typically aimed

at achieving certain design goals. The self-explanatory *enemy out of nowhere* pattern can be implemented with different setups, such as with a hidden block, by placing enemies just outside the visible area, and so on. In fact, a key difference between a pattern and a setup is that the latter is concrete, whereas the former represents an idea that can be realized in different ways; thus, a setup may often be used in different patterns.

1.2.4 *Genre*: A characterization of levels that use certain mechanics, setups, and patterns. For example, *Kaizo* levels are characterized by punishingly strict setups such as shell jumps [28]. *Troll levels* use setups such as death doors, misleading indicators, and enemies out of nowhere that will "troll" the player. *Music levels* use sophisticated setups to produce custom music as the player progresses through the level.

2 Outputs: are classified in *player-dependent* and *player-independent*.

2.1 Player-dependent outputs are those that can only be assessed relative to a player with specific skills, taste, or inclinations. This does not mean that they are entirely subjective, but their "scale" should be adjusted for each player.

2.1.1 *Aesthetic*: The visual appeal and "vibe" of a level, achieved through decoration, theme choice, and repurposing items to look like something else. An example of purely aesthetic decoration is *pixel art*, which refers to arrangements of blocks without any functional purpose to improve the level's visual appeal; for example, a Zelda-themed level may use pixel art to draw the Triforce or Link's Master Sword.

2.1.2 *Clarity*: How easily a player can navigate a level and understand how it should be played, which mechanics should be used, and ultimately the maker's intent. Clarity is player-dependent because players with experience will generally be able to "read" a level more easily than novices. A level's clarity directly affects its *learnability* but is generally independent of its *difficulty*; for example, a complicated and obscure level that requires *memorization* has low learnability, even if it may still generally be easy to complete in terms of required skills.

2.1.3 *Difficulty*: The amount of skill required to successfully clear a level. For example, a *punishing* level is one with extreme difficulty; while this makes it accessible only to highly skilled players, a harsh difficulty can still be enjoyable provided it is introduced in a *fair* way.

2.1.4 *Sentiment*: A player's overall feeling, impression, and enjoyment of a level. This is the ultimate player-dependent assessment: whereas all other output categories concur to determining whether playing a level produces positive or negative sentiments, it is still possible that a level checks all the boxes but still is not found to be enjoyable by some players. In contrast, a level that is *fun* to play is arguably an example of valid design, even if it flouts conventions or subverts expectations.

2.2 Player-independent outputs are those that can be assessed uniformly for all kinds of players.

2.2.1 *Complexity*: The sheer length, density of information, number of moving parts, and the intricacy of the setups and patterns deployed in a level. A level with high complexity is not necessarily difficult: for example, an *auto level* (which moves the player through the level to the goal without requiring user inputs) has high complexity but is trivial to play. Conversely, the designation *short and sweet* characterizes levels with low complexity but high-quality, enjoyable design.

2.2.2 *Consistency*: The extent to which the parts of a level are built in a coherent way, that is, they have similar or complementary styles and fit well together. For example, *little Timmy levels* (chaotic, low-effort levels with gratuitous item spamming) are characterized by low consistency, and so are levels with a *description/genre mismatch*—for example a level that claims to be a speedrun, but then has no tight timing constraints. In contrast, classic Miyamoto design (or what `PangaTAS` calls "the formula"[17]) is characterized by high consistency.

2.2.3 *Creativity*: The *novelty* of a level's concept, for example in the use of original game setups and mechanics that combine game items in unintended, surprising ways.

2.2.4 *Fairness*: The degree to which a failure can be attributed to the player's lack of skill rather than being unavoidable, random, or the result of (deliberate or unintentional) lack of clarity in the design. For example, a *troll* is a pattern that is deliberately, and often punishingly, unfair. However, a Kaizo level can be punishingly difficult and yet fair. Levels that use lots of undirected *randomness* are often unfair, as even the most skilled player needs a favorable "roll of the dice" to be able to complete the level. This also explains why fairness should be considered player independent.

2.2.5 *Idea implementation*: How effectively a concept or idea is realized in the concrete design of a level. The acronym *CCBE* ("Cool Concept, Bad Execution") is often used to judge levels where a solid concept is undermined by poor implementation.

2.2.6 *Pace*: The rhythm and "tempo" of a level, which follow its temporal/dynamic constraints. The related concept of *flow* characterizes levels with an enjoyable, natural pace. To a certain extent, pace depends on layout: for example, a certain arrangement of enemies would force a fast pace. A fast pace often increases a level's difficulty, by requiring tight reaction times.

2.2.7 *Robustness*: How "proofed" against improper or unintended usage a level is. A robust design prevents players from getting softlocked or cheesing (skipping) challenges. In contrast, the notion of *jank* denotes setups that are clunky, glitchy, and rarely work as intended. We consider robustness player independent because it should be evaluated "adversarially", ideally in a way that covers all possible player interactions.

2.2.8 *Strictness*: The narrowness of the "intended path" within a level or its timing. High strictness means there is

[17] https://thechonker.fans/#formula

only one exact way to succeed with no room for error. A *claustrophobic* level is one that is physically and unpleasantly strict.

B. Threats to Validity

Internal validity. The process we followed to select and extract data from YouTube video may have produced a *biased* sample, not fully representative of the concepts that characterize design quality in SMM2. Our methodology tried to mitigate this threat in different ways: *i*) We selected popular streamers whose content specializes in SMM2, who have a long track record and "cred" in the Mario Maker and Kaizo ROM hacks communities; *ii*) We manually validated, on a smaller sample, the reliability of the LLM-based segment extraction and topic clustering, before applying it to all 500 videos; *iii*) We performed a posteriori validation of the taxonomy (Sec. III-C), which introduced additional concrete elements. Despite these precautions, we do not claim that our taxonomy is *complete*—especially for what concerns concrete elements. However, it should capture the most important and prominent factors that characterize quality in SMM2 level design, and it can be a useful basis for further extensions or specialization.

We did not perform respondent validation: a process in which research results (i.e., our taxonomy) are submitted to the originators of the sources (i.e., the streamers) for their validation. Had we misinterpreted the streamers' commentary or missed their meta-context, it may have introduced threats to credibility and confirmability of our results. In this instance, the authors' domain expertise served as a mitigation strategy.

While we tried to be as precise as possible, the definitions of abstract categories in our taxonomy are not formal, and hence their applicability to new concrete elements may sometimes be approximate or judgment-dependent. For example, the distinction between *setup* and *pattern* may be fuzzy in some cases. This is an expected result of a research methodology that is mainly qualitative [2]. Nevertheless, the concepts presented by the taxonomy remain generally clear and straightforward to apply to new elements; in fact, a qualitative analysis is a necessary precursor step to developing a fully rigorous, and possibly quantitative, understanding of an unexplored domain.

External validity. A possible bias affecting the *generality* of our taxonomy comes from the fact that all streamers (whose videos we used as data) are *expert* players; hence, it may not fully capture the views on design quality held by novices. On the one hand, expert players are more likely to hold complex and nuanced views about level design; thus, to some extent, a certain experience level is needed even to be able to fully appreciate design choices, and to clearly articulate opinions about them. On the other hand, the videos we analyzed feature a wide variety of levels—in terms of both design quality and difficulty; hence, our focus on expert players did not prevent us from exploring the design landscape far and wide.

Strictly speaking, Fig. 1's taxonomy only applies to SMM2 levels. However, given that SMM2 shares a lot of features with its predecessor SMM1, and that all streamers we considered were also active players of SMM1 before SMM2 was released, it is very likely that our taxonomy largely works for SMM1 level design too. It could also be the basis of future work extending and revisiting it to cover the design concepts characterizing the levels produced by users of other popular 2D platformer level editors, such as those for Kaizo ROM hacks (*Lunar Magic*), *Super Meat Boy*, *The End is Nigh*, *Celeste*, and *Levelhead*.

V. Discussion

Let us briefly discuss how our taxonomy of level design quality in SMM2 fills some knowledge gaps and can be the basis of future work in this area.

To our knowledge, our taxonomy is the first to make an explicit distinction among different contraptions—in particular between *setup* and *pattern*: while both are combinations of design ingredients, a setup is more specific and concrete, whereas a pattern is more general and abstract. While Sec. IV-B acknowledged that the distinction between the two still is occasionally fuzzy, it is not arbitrary, and reflects two genuinely distinct concepts. This distinction is helpful to clarify previous literature about level design, which often used the term *pattern* with a very broad meaning [11], [12]. For example, Thompson observes [11] that "design patterns can occur at varying levels of abstraction" and "it is important to distinguish specific patterns even if they could be perceived on a more abstract level as the same". Our distinction between setups and patterns naturally captures such aspects; in fact, Thompson's catalog includes both a few concepts that we would characterize as patterns according to our terminology (e.g., risk-reward) and several more concepts that fall under our definition of setup (e.g., single-enemy, flooded-gap, spike-valley).

In Sec. II, we noticed how most monographs and books on (level) game design present principles that are not specific to (2D) platformers—and thus may overlook the specificity of this kind of games, or fail to account for the diversity of design styles that are available in user-produced content. Our taxonomy provides vocabulary and concepts that can be used to bridge this gap. As an example, consider Salmond's principle of level design #10: "Good level design is driven by mechanics" [6]. The principle's description does not refer to any platformer examples, but several of our taxonomy's concepts are clearly relevant, such as: *i*) level design realizes a relation between *input* and *output*; *ii*) the vital role of communication between designer and player, which affects *clarity*, *consistency*, *robustness*, *strictness*, and even *fairness*; *iii*) the observation that *items*, (game) *features*, and *mechanics* are sometimes changed during level design is applicable only in a narrow sense to SMM2 level design, given that a maker can decide which elements to use but cannot change anything in Nintendo's game engine.

The concepts in our taxonomy could also be used as foundation to ferret out the relation between inputs and outputs in a level. Take, for instance, a level's *difficulty*: in SMM2 there are metrics, such as the clear rate or Nintendo's difficulty rating, that plausibly capture difficulty as experienced by players [1]. A natural follow-up question is: what contraptions most affect difficulty? Some contraptions, such as a *shell jump* or certain

kinds of *spike jump* setups, are strong determinants of difficulty, as they require very precise player inputs and go well beyond what is normally present in a mainstream platforming level. In contrast, other contraptions can feature in both easy and hard levels; for example, the *damage boost* mechanic may be optional (thus lowering the difficulty by allowing an additional hit) or required (thus increasing the difficulty by demanding a quick reaction). By the same token, a *troll* is *unfair* by definition; however, it may still produce a positive *sentiment* if it is part of a well-designed troll level or if it is used sparingly to provide a further element of challenge (in fact, a few small trolls that keep the player on their toes are part of the Kaizo genre since the seminal "Kaizo Mario World"[18]).

VI. Conclusions

We presented a taxonomy of design quality for SMM2 levels, which we built primarily based on the commentary produced by popular YouTube streamers while playing SMM2 levels. The taxonomy includes concepts related both to the input of the design process (e.g., which mechanics or items are used) and to its output (e.g., how difficult or creative the level is). The taxonomy's abstract concepts provide a custom vocabulary to describe the design process that covers the broad variety of styles, difficulty, and quality that is encountered in the millions of user-produced SMM2 levels.

References

[1] C. A. Furia and A. Mocci, "What makes a level hard in Super Mario Maker 2?" in *IEEE Conference on Games, CoG 2025, Lisbon, Portugal, August 26-29, 2025*. IEEE, 2025, pp. 1–8.

[2] J. Corbin and A. Strauss, *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*, 4th ed. SAGE Publications, 2015. [Online]. Available: https://doi.org/10.4135/9781452230153

[3] G. Smith, M. Cha, and J. Whitehead, "A framework for analysis of 2D platformer levels," in *Proceedings of the 2008 ACM SIGGRAPH Symposium on Video Games*, ser. Sandbox '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 75–80. [Online]. Available: https://doi.org/10.1145/1401843.1401858

[4] J. Feil and M. Scattergood, *Beginning game level design*. Thomson Course Technology, 2005.

[5] C. W. Totten, *An Architectural approach to level design*. CRC press, 2019.

[6] M. Salmond, *Video game level design: how to create video games with emotion, interaction, and engagement*. Bloomsbury Academic, 2021. [Online]. Available: https://doi.org/10.5040/9781501380907

[7] A. Canossa and G. Smith, "Towards a procedural evaluation technique: Metrics for level design," in *The 10th International Conference on the Foundations of Digital Games*. sn, 2015, p. 8.

[8] K. Hullett and J. Whitehead, "Design patterns in FPS levels," in *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, ser. FDG '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 78–85. [Online]. Available: https://doi.org/10.1145/1822348.1822359

[9] G. Smith, R. Anderson, B. Kopleck, Z. Lindblad, L. Scott *et al.*, "Situating quests: Design patterns for quest and level design in role-playing games," in *Interactive Storytelling*, M. Si, D. Thue, E. André, J. C. Lester, T. J. Tanenbaum *et al.*, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 326–329.

[10] A. Khalifa, F. de Mesentier Silva, and J. Togelius, "Level design patterns in 2D games," in *2019 IEEE Conference on Games (CoG)*. IEEE Press, 2019, p. 1–8. [Online]. Available: https://doi.org/10.1109/CIG.2019.8847953

[11] T. Thompson, "The fine line between rehash and sequel: Design patterns of the Super Mario series," in *Design Patterns in Games Workshop*, 2015. [Online]. Available: http://dpg.fdg2015.org/

[12] S. Dahlskog and J. Togelius, "Patterns and procedural content generation: revisiting Mario in world 1 level 1," in *Proceedings of the First Workshop on Design Patterns in Games*, ser. DPG '12. New York, NY, USA: Association for Computing Machinery, 2012. [Online]. Available: https://doi.org/10.1145/2427116.2427117

[13] A. Summerville, J. R. H. Mariño, S. Snodgrass, S. Ontañón, and L. H. S. Lelis, "Understanding Mario: an evaluation of design metrics for platformers," in *Proceedings of the 12th International Conference on the Foundations of Digital Games*, ser. FDG '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3102071.3102080

[14] N. W. Eloe and T. C. Meyer, "Indicators in Super Mario Maker 2: Evolution and rhetorical signification in visual languages," *J. Vis. Lang. Comput.*, vol. 2022, no. 2, pp. 12–25, 2022.

[15] C. Lewis, J. Whitehead, and N. Wardrip-Fruin, "What went wrong: a taxonomy of video game bugs," in *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, ser. FDG '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 108–115. [Online]. Available: https://doi.org/10.1145/1822348.1822363

[16] B. Aytemiz and A. M. Smith, "A diagnostic taxonomy of failure in videogames," in *Proceedings of the 15th International Conference on the Foundations of Digital Games*, ser. FDG '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3402942.3402979

[17] M. A. M. Cuerdo, A. Mahajan, J. Mao, and E. F. Melcer, "Try again?: A macro-level taxonomy of the challenge and failure process in games," in *2023 IEEE Conference on Games (CoG)*, 2023, pp. 1–8.

[18] M. A. M. Cuerdo, A. Mahajan, and E. F. Melcer, "Die-r consequences: Player experience and the design of failure through respawning mechanics," in *2021 IEEE Conference on Games (CoG)*, 2021, pp. 01–08.

[19] M. A. M. Cuerdo and E. F. Melcer, ""I'll be back": A taxonomy of death and rebirth in platformer video games," in *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, ser. CHI EA '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1–13. [Online]. Available: https://doi.org/10.1145/3334480.3382863

[20] S.-K. Thiel and P. Lyle, "Malleable games - a literature review on communities of game modders," in *Proceedings of the 9th International Conference on Communities & Technologies - Transforming Communities*. New York, NY, USA: Association for Computing Machinery, 2019, p. 198–209. [Online]. Available: https://doi.org/10.1145/3328320.3328393

[21] H. Duan, Y. Liu, and W. Cai, "User-generated content and editors in games: A comprehensive survey," *ACM Trans. Multimedia Comput. Commun. Appl.*, Feb. 2026, just Accepted. [Online]. Available: https://doi.org/10.1145/3798165

[22] A. Singh, A. Fry, A. Perelman, A. Tart, A. Ganesh *et al.*, "OpenAI GPT-5 system card," 2025. [Online]. Available: https://arxiv.org/abs/2601.03267

[23] S. Mandal, N. Gupta, A. Talewar, P. Ahuja, P. Juvatkar *et al.*, "Idpleaderboard: A unified leaderboard for intelligent document processing tasks," https://idp-leaderboard.org/, 2025.

[24] K. Enevoldsen, I. Chung, I. Kerboua, M. Kardos, A. Mathur *et al.*, "MMTEB: Massive multilingual text embedding benchmark," 2025. [Online]. Available: https://arxiv.org/abs/2502.13595

[25] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," 2020. [Online]. Available: https://arxiv.org/abs/1802.03426

[26] R. J. G. B. Campello, D. Moulavi, A. Zimek, and J. Sander, "Hierarchical density estimates for data clustering, visualization, and outlier detection," *ACM Trans. Knowl. Discov. Data*, vol. 10, no. 1, Jul. 2015. [Online]. Available: https://doi.org/10.1145/2733381

[27] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu *et al.*, "Judging LLM-as-a-judge with MT-bench and Chatbot Arena," 2023. [Online]. Available: https://arxiv.org/abs/2306.05685

[28] J. Bycer, *Underground Design of Kaizo Games*. Cham: Springer International Publishing, 2024, pp. 1925–1929. [Online]. Available: https://doi.org/10.1007/978-3-031-23161-2_379

[18] https://en.wikipedia.org/wiki/Kaizo_Mario_World