The Creative Design-Engineer Divide: Modular Architecture and Workflow UX

Brian Packer[1] [0000-0003-2105-0042], Simeon Keates[2] [0000-0002-2826-672X], Grahame Baker[1]

[1] University of Greenwich, Chatham Maritime, ME4 4TB, UK
[2] University of Chichester, Bognor Regis, PO21 1HR, UK
B.W.Packer@greenwich.ac.uk

Abstract. There are competing priorities between creative freedom and the need for robust, stable software frameworks to facilitate the rapid implementation of creative ideas in game development. This may result in a disparity between system and user requirements. Qualitative data extracted from seminars at the Game Developers Conference informs the design of several interviews with veteran game-system designers to explore this phenomenon. A survey of modular software plug-ins from the Unity Asset Store then validates the interview findings and explores the benefits of modular software architectures. Findings indicate that modifications to the native user experience (UX) design of Unity and plug-ins that reengineer for different workflows are most popular. The most popular workflows provide for data, asset, and project management. Discussion reflects on how modular architecture can alleviate points of failure within a game engine's architecture whilst providing customized usability for different user needs.

Keywords: Game design, workflow, user experience, qualitative analysis

1 Introduction

Video game development stands at the intersection between the disciplines of creative design, system design and software engineering [1]. The software used to develop games must provide a robust framework by which ideas and content can be rapidly implemented. So-called "game engines" (also called "Integrated Game Development Environments" (iGDEs), so as to not conflate the development software suites with the extrinsic (primarily graphical rendering) libraries that compile and execute a game) must provide a flexible software architecture to facilitate this with maximum technical accessibility for less technically trained system/art designers.

The artistic nature of games demands that neither limit nor consensus be enforced on what defines "a game" [2] [3], yet this poses a unique challenge to the design management of such software and the traditional requirements gathering process for identifying how to deliver an intuitive user experience (UX) for uninhibited implementation. Furthermore, developing video games is a multi-discipline task, demanding a range of artistic and technical skills which may benefit from varied UX needs [4].

Until the emergence of more purposefully accessible "all-in-one" visual editors in the early 2010s [5], common practice was to build toolchains to interface third-party tools and bespoke development environments (figure 1). But the advent of, and shifting publication/access rights to, engines like Unity and Unreal have opened the discussion of whether it is better to build an engine for a game or build a game using engines that comes with much of the engineering precompiled and abstracted.

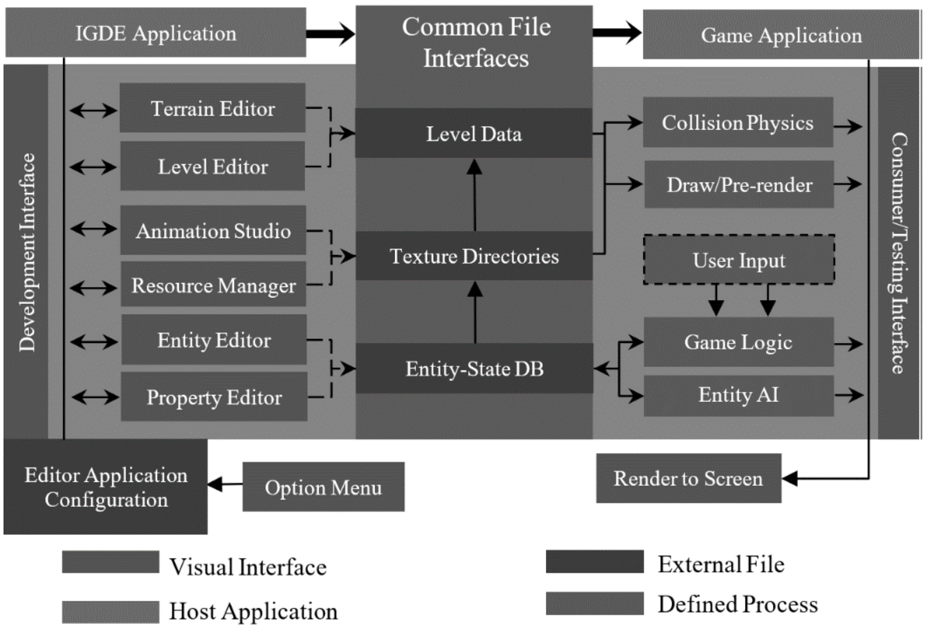

Fig. 1. An example architecture of a so-called "Game Engine", highlighting the distinction between the iGDE and Game presentation layers.

This investigation sought to profile 3 sources of qualitative data from game designers relating to tool engineering by using content analysis. Whilst each source has limits in isolation, cross-referencing 3 distributed sources to inform generalized findings about tool engineering was thought to allow for a better understanding of the specific modules game designers most need engineers to reflect upon. The first analysis looks at seminars presented at the Game Developers Conference, the second analyzes transcript data from interviews with 3 veteran game system designers and the third is a survey of the tools and plug-ins commercially available on the Unity Asset Store.

2 Game Developers Conference Content Analysis

Prior requirement analysis derived from content analysis of "Game Tool" case-studies indicated that tool engineering with a focus on quality assurance (QA) and iterative design was a major concern amongst both game designers and tool engineers on small to medium-sized development teams [6]. The findings were generalized and only identified broad trends between disciplines of game developers and different studio structures. This prompted further investigation to better identify specific functional requirements and possibly identify aggregate trends in game engine design management. Replicating these methods and cross-referencing for either tooling or production key-phrases resulted in some indicative observations, including:

- “Editor” was the most repeated key word, appearing 8.9 magnitudes (***V***) of standard deviation above the average frequency (5.1+6.3***V***) across all 701 meaningful key words or phrases. This indicates it is the “inductive generic”, the word most descriptive of the sample.
- Tool-engineering key phrases: "Script(ing)" (μ+7.9σ) and "Code" (μ+1.7σ), "Data" (μ+5.1σ), "Animation" (μ+4.3σ) and "Kinematic" (μ+2.9σ) and "Modular" (μ+1.8σ), were also notably repeated above average across the sample.
- Iterative design and QA key phrases included "Iteration", "Bug", "Debug(ging)", "QA" and "Test", and these made up 33.2% of all key phrase distribution.
- Collectively, the sample of key phrases relating to users, user-experience, creativity, productivity, and workflow comprised 36.3% of all key phrases.
- Some seminars were in the context of utilizing general-purpose game engines (Unity, Unreal, Frostbite) combined with bespoke modules/libraries to support their specific design requirements. Other seminars discussed entirely proprietary (“in-house”) game engines or bespoke, self-contained, and automated tools built to support the development, or even procedural generation, of specific games.

It is difficult to validate generalized knowledge from one sample of content analysis. These key phrases could, however, be used to inform the design of further investigation. Combined, each investigation could then be cross-referenced with these preliminary findings to build a better representative aggregate of user needs.

3 Industry Interviews

Triangulation is the process of sampling multiple relevant data sources and cross-referencing the findings of each analysis. This can be used to confirm theoretical models and add detail to those models [7] [8]. Since, by nature, archival footage cannot provide elaboration on any findings observed, interviews with suitably experienced participants may provide stronger evidence to inform better design of game tools.

3.1 Interview Design

Interview questions were designed to use sentiment-evoking keywords to pre-contextualise the participant’s answers towards different categories of observation, as described in table 1. Responses could then be codified as independent variables [9]. The selection criteria of the candidates can be considered multiple factors of linear regression, with the most common sentiment mined from semantic coding of each response serving as a dependant variable which can be measured. Participant selection criteria act as coefficients of regression, adding to the experiment power, meaning a small number of interview participants can be used to strongly contribute to the observable criteria for a third investigation [10] [11]. The selection criteria for participation included:

1. Primary discipline of system designer (as opposed to Artists)
2. AAA Studio and Small-Medium Enterprise (SME) studio work experience
3. Professional experience shipping a game using a proprietary engine.
4. Professional experience with either Unreal or Unity engines.
5. Worked on the development of at least 1 released game either operating as a service for at least 5 years or perpetual open-access development for 5 years.

Table 1. Open-ended interview prompts and their respective semantic context-clues.

Q1.	What technical limitations have you found most limiting when trying to implement new ideas?
Semantic Keywords: Technical, limitation, implementation, ideas	
Q2a.	What do you feel is the biggest loss in efficiency, sometimes referred to as bottlenecks, in your development and iteration process?
Semantic Keywords: Efficiency, iteration	
Q2b.	What do you feel is the biggest loss in efficiency in training new designers?
Semantic Keywords: Efficiency, training, designers	
Q3a.	With 7 being high and 1 being low, what impact do you think this issue has on your creative expression?
Semantic Keywords: Creative expression	
Q3b.	(Optional) What are your thoughts of contextual development interfaces?
Semantic Keywords: Context-sensitive, interfaces	
Q4.	What does creativity mean to you as a systems designer?
Semantic Keywords: Creativity, system design	

The reason system designers were selected over artists is due to the lack of constraints on the scope of what system designers do within game design. It was thought this would give the broadest perspectives on many tools, rather than the specialized tools different disciplines of artists tend to use. The need to have worked at both AAA and smaller studios was informed by the findings that tool needs and priorities differed depending on the expectation that a studio will have more general-purpose roles or the scope for large-staffed dedicated departments. Experience working with both proprietary and modular, general-purpose engines helped to control the bias towards one or the other. Finally, the requirement to have such extensive development experience on a single game was to control for participants who may have used tools still in early-development or not reasonably functional for fair measure.

3.2 Interview Analyses

After removing interviewer interjection or clarification there was 47 minutes and 35 seconds (2855 seconds) of participant data at an approximate average rate of 2.67 words per second. Continuing to replicate the methods used in the Interpretive Content Analysis model; Layers 1, 2 and 3 (Literal, Contextual and Observational) were supplemented using 3 methods of content analysis: Lexicography, Coding and Distillation. Layer 4 (Meta-factor Analysis) was predetermined by the design of the interview questions and participant criteria.

Lexicographical Analysis. There are some limitations to using the same word-pair analysis used on the GCD data for interviews. The open nature of the interview questions resulted in more topical variance than GCD seminars. This, combined with the smaller sample of word data, means the data distribution was weaker with fewer impactful trends. Furthermore, the context-clue keywords from the question design had to be removed from the data to avoid inflating their value: “Technical”, “Ideas”, “Limitation”, “Implementing”, “Efficiency”, “Iteration” and “Context” are "leading" words. The transcriptions removed the interviewer's speech to further mitigate this effect. Words were grouped by how many magnitudes of standard deviation they appear above the average (table 2).

Table 2. Frequency ***(f)*** of which key words or word-pairs occurred in the interview transcripts, grouped by prominence as determined by magnitudes of standard deviation from the mean.

Very Prominent ($f >= \mu+3\sigma$)				***More Prominent*** ($f >= \mu+2\sigma$)		***Prominent*** ($f >= \mu+1\sigma$)			
Key Phrase	***f***	**Key Phrase**	***f***	**Key Phrase**	***f***	**Key Phrase**	***f***	**Key Phrase**	***f***
Game	59	New	36	Interesting	19	Space	14	Character	12
Engine	52	Make More	27	Context	19	Working	13	Animation	11
Tool	46	Time	26	Creativity	18	Developer	13	Level	11
Work	41	Team	24	Feel	17	Unity	12	End Up	10
Different	39	Designer	23	WoW	17	Object	12		

"Game Engine", "Engine" and "Tool(s)" are again a primary focus by a large margin. This is reflective of the GDC findings and highlights concurrency between data. Strong focus on "team" and "designers" indicating a level of generalization reflective of experienced collaborative designers. "Making more" and "time" reflect the main metrics by which productivity is measured. "Context" and "Feel" were almost exclusively used in the context of tool user experience. The phrase "Space" lacks context in abstraction, but reviewing the source data showed that it was often used as short-hand for "3D Space", "head space" (sic) and "work space" (sic). "3D space" references usability or functionality within a level or scene, whereas "headspace" and "workspace" allude to concepts of user experience or productivity [12] [13], whilst “Object”, “Character”, “Animation” and “Level” each represent different workflows for game designers (Level Design, Animation and Character, and System Design). Finally, the phrase "end up" typified a sentiment of resignation with systems that do not work as expected, or desired, but can be used imperfectly to achieve a goal.

Finally, “WoW” was used as short-hand for the MMORPG “World of Warcraft” and was the context for a bespoke game engines with a relatively long product life cycle, 18 years, maintained for a single game-as-a-service. For comparison, Unity is mentioned slightly less and represent a more modern modular engine. Unreal was rarely mentioned.

Interview Coding. Coding is the process of assigning meaning to qualitative data in a systematic way. Distillation is the process of recursively categorising semantic data into less discrete groups of data, inferred from commonality between the meaning of key phrases [14]. The data was reviewed, and timestamps placed denoting interview structure (breaks, questions, clarifications) and subject categorisations (figure 2).

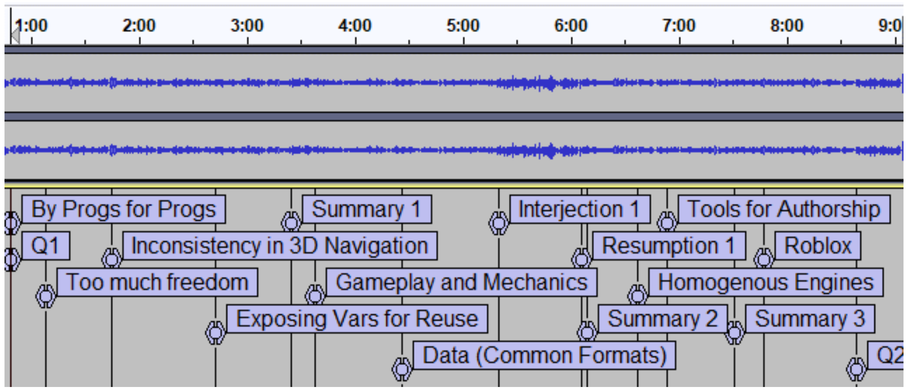

Fig. 2. Example of the coding process, starting with cataloguing the raw interview audio data labelling and chaptering.

These codes were then assigned to 1 of 5 categories and given semantic tags. For example, “3D Navigation” may be the subject, but this may be discussed in the context of the task of prototyping “3D Block-Outs”. If a specific feature was discussed in this context, it was noted, with either affirmative (positive) or contradictory (negative) statements assigned to the given statement [15]. Verbatim word-association was used to divide uninterrupted participant responses into discrete “chunks”. This is known as “open coding” and allowed the most accurate model representation of the interview data (table 3).

Once “open coding” was complete, a process of abstraction called “recursive coding” sought to consolidate semantically similar codes into broader groups. These chunks are then consolidated and measured for reoccurrence to give a better measure of the weight assigned to topic across all participants. Continuing with the previous example, multiple participants may discuss specific problems with “3D Block Outs” but they may each discuss different tools or aspects that clarify their responses. An aggregate of the task category would come under the discipline of “Level Design”, whereas the tools may not semantically relate (table 4).

Table 3. An extract of the table used during the “open coding” stage of interview reviews. Each row is self-contained by code category. Each column represents each expression of sentiment.

Task	3D Block-Out	Pre-Art Implementation	Rapid Implementation	Reengineering Assets	…
Affirmative Sentiment	Abstract Proportions	Fast Concept Testing	Consistent UI, by context	“Pro-mode” 3D Designer UI	…
Contradictory Sentiment	Too much freedom	Features prioritised over UX	Abstract Relative Scaling (2D/3D)	Snapping Overrides	…
Experience or Reflection	Inadequate requirements analysis	Requirements vary by discipline	Too much access to incidental vars.	Not enough accessibility to data structures	…
Feature Highlight	Relative Scaling	Snapping	Sandbox Testing	Prefabrication	…

Table 4. An extract of the “selective coding” stage of interview distillation.

Task	Level Design	System Design	System Design	System Design	…
Affirmative Sentiment	User Familiarity	System Testability	User Profiling	User Profiling	…
Contradictory Sentiment	Data Accessibility	User Profiling	Navigation Accessibility	Navigation Accessibility	…
Experience or Reflection	User Profiling	User Profiling	Data Accessibility	Data Accessibility	…
Feature Highlight	Controls	Controls	Prototyping	Prototyping	…

Given coding is an intermediary step between literal and interpretive analysis, the results expectedly reflected the lexicographical analysis. Each category of coding had between 79 and 96 chunks assigned across all interviews. *Affirmative sentiments* were focused on designing tools for workflows and “contextual interface” design (47%). *Contradictory sentiments* were split between two smaller trends, “data accessibility”

(24%) and “collaborative design” (17%). “Data abstraction” tools, including scripting and data visualization, were the most discussed *feature highlights* (37%).

Interview Distillation. Much debate has been presented across most scientific fields engaged with any analysis of naturalised (human) data about the validity, and practicality, of presenting comprehensive depictions of nuanced user behaviour and opinions [16] [17] [18]. The more data is abstracted from a verbatim transcription, the more an observer’s “theoretical priori” orientation may influence their neutrality [19]. With due consideration to this, some summary findings from the interviews are presented here to provide context for the data previously presented in abstraction, taken verbatim where possible:

Question 1. On technical limitations on creative development:

- Engines are built “by programmers, for programmers”, with too much access to variables without the ability to expose only what is meaningful.
- There is “inaccessibility to common tools/formats [in favour of proprietary ones]”.
- “All in wonder” tools (like Unity) limit accessibility by making anything possible only once you know how to setup and build the underlying game systems.
- Technical limits on live collaboration prevents consistency in design intent between disciplines.

Question 2. On losses of efficiency, both in a HCI and team-work context:

- Lost context, or noisy UIs, break the comfortable and intuitive headspace of design task. This can overwhelm new users, distract experienced users, and slow down proficient users.
- Learning how to do a new task is a huge limit. Especially in proprietary engines where documentation is limited, and enquiries cannot be crowdsourced externally.
- Game data concurrency, especially in the context of live collaborative development, can throttle the iterative cycle of design, implement and testing game “feel”.
- Repetitive or chain-tasks often require specific replication of inputs or task sequences. Losing interface focus during these sequences often means starting over.

Question 3. The impact on creative expression returned a mean score (μ) of 6 out of 7, with a standard deviation (σ) of 1.4, suggesting general agreement that responses to question 2 heavily impact creativity.

Question 4. On the definition of creativity from the perspective of system design.

- Creativity is working within the constraints of a system(s) to develop novel application of, or interactions between, those system(s).
- Creativity is about communicating a system to a player through the design and presentation of data in a way that is immersive and accessible.

4 Unity Asset Store Survey

Most content on the Unity Asset Store provides prefabricated art resources, not functionality-adding components. A quick analysis of that repository provided no unexpected results. 3D art was more prominent than 2D art, with Sound and Visual Effects (VFX) being considerably less in supply (figure 3).

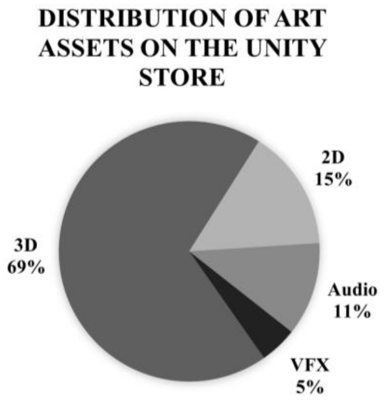

Fig. 3. Distribution (%) of art plug-ins on the Unity Asset Store, by art sub-disciplines.

Tools and plug-ins (modules) are a distinct category unto themselves and are further sub-divided by a moderated tagging system. These tags tend to reflect distinct workflows or disciplines within game development, though some describe specific tasks or functionality standard to most game engines. The distribution of these plug-ins provides much more relevant data to the design of engine architecture than the artistic plug-ins, as well as specific (and independently assigned) semantic grouping.

Table 5. Distribution of the sum number of plug-ins ***(f)*** in each category.

Category	***(f)***	**Category**	***(f)***	**Category**	***(f)***
Utilities	1638	Game Toolkits	315	Sprites Control	209
GUI	1248	Animation	314	Level Design	150
Integration	1104	AI	297	Visual Scripting	145
Particles & Effects	593	Network	270	Video	89
Input Management	455	Camera	263	Localization	78
Physics	387	Audio	257	Painting	49
Modelling	385	Terrain	234	Version Control	18

There were several categories listed separately to the tool/plug-in that bear inclusion:

- Templates (2774 (*f*) plug-ins), which includes precompiled Unity project structures, data configurations, databases which support certain common game systems, tutorial projects for learning to use Unity, and finally, resource packs for certain themes of game.

- Services (21 (*f*) plug-ins), which mainly included plug-ins for connecting game systems and interfaces to financial transaction services or instant messaging APIs.
- Machine Learning (17 (*f*) plug-ins), which provide a variety of neural net and competitive agent libraries for programming AI for games.

Across all tool categories, “Utilities”, “GUI” (including Unity GUI Managers) and “Integration” tools were the most prominent categories (table 5), occurring 2.9, 2.0 and 1.7 standard deviations above the mean frequency of all tool categories, respectively, which was concurrent with data gathered from the interviews.

Whilst “Integration” has clear relevance to the software engineering and compatibility aspects of game development, “GUI” could refer to both the implementation of GUIs into games, as well as GUIs for the Unity Engine. Regardless, GUI still has a clear association relative to an aspect of game-design. “Utilities” is a semantically vague term, despite it being (speculatively, causing it to be) the largest category of plug-ins. Deeper investigation of the “utilities” category was carried out by sampling the 50 most reviewed 5* plug-ins. Whilst this could not give a representative semantic assignment to the category, it could provide data relating the most used and positively received plug-ins. This can be used to complete the triangulation of the GDC seminar data and the interview data. Some observations included:

- 80% provided modular GUIs to provide functionality not native to Unity.
- 63% re-engineered GUI elements native to Unity or extended/overwrote the functionality of existing Unity interfaces or tools.
- 28% provided interfaces or functionality for managing, reviewing, or controlling game data and data connected to game-assets. 4% were script-driven.
- 60% delivered work-flow enhancement for a given task or discipline. 24% specifically cited increasing developer productivity as a feature.
  - Examples included interfaces or scripts adding functionality to improve workflow for tasks including: pooling and asset inspection (9), programming (6) level design (5), debugging (5), security and data obfuscation (4), and quality assurance, texturing, animation, and particle effect management (1).
- 54% provided methods and interfaces for game project file and asset management, particularly for optimization.
- 28% provided functionality or interfaces for optimization of render and compute performance, primarily through asset dependency calculation and asset pooling, though some provided for the implementation of level-of-detail control on art.
- 6% provided tools for procedural generation for either 2D or 3D level design.

Reviewing rating to price ratio found that, of the 100 most expensive plugins in each category (including any plug-ins of equivalent price to the price floor), 36-37% of each of the 3 top categories did not have ratings, indicating insufficient reviews and (implicitly) sales. Utilities had the most favorable ratings (62.7%, 4-5 star ratings), but favorability was level across all 3 categories (62.7%, 56% and 57%, respectively).

Notably, quality assurance (excluding performance optimization) was a focus of very few plug-ins; 1 in the sample of most popular tools, and 10 across all tool categories. The "Testing" tag also yielded only 34 results. This is less than anticipated given the relative focus on QA from SME developers at GDC.

Finally, in two samples taken 8 months apart (October 2020, June 2021), there was a significant reduction in the number of System Templates (-24%, 878 to 667). Filtering for version-compatibility shows that much of this reduction came from deprecated support due to versioning. This highlights that the problems with legacy systems, alluded to in the interviews, are not isolated to long-life proprietary systems/engines.

5 Conclusions

To summarize, across 3 investigations into different sources of game system designer behavioral patterns and perspectives, there was repeated evidence in favor of designing game engine user experiences that favor contextual design and optimizes for discipline-aligned workflows.

The first analysis indicated that Editor design was the most referenced topic across professional seminars reflecting on tooling or production issues. Scripting, Data, Animation and Modular (design) were key tasks of focus for tools, but phrases connected to concepts of productivity, users and usability, creativity and workflow were the most prominent phrases across all seminars.

The second analysis supported these findings and incorporated them into designing interviews with seasoned game system designers. Those interviews emphasized contextual design for given tasks is greatly preferable and that exposing too much data to the point of over-accessibility is destructive to the user experience of any engine or tools, with the caveat that controlling what is or is not exposed is preferable to not being able to access essential data under any circumstances.

The third analysis highlighted the tools and utilities most used and reviewed by game designers using the Unity Asset Store. These plug-ins largely override the functionality and user experience of the Unity engine in favor of workflows optimized for given tasks or aspects of game development. The most common of these was project and data/asset management, primarily for performance optimization and project refactorization.

Modular architecture is common in both proprietary and general-purpose game engines as it allows for the agile assessment of the game system designer's needs. When development of these modules can be aggregated across larger audiences (such as the Unity Asset Store) there is an almost evolutionary “survival of the fittest” effect that delivers enhanced usability. However, data management and abstraction are major restrictions on meeting these needs, and further research is needed to understand how data and creative design can be better interpolated to free up experienced designers and increase accessibility to initiate designers.

References

1. Lightbown, D.: Designing the User Experience of Game Development Tools. 1st edition 1st edn. A K Peters/CRC Press (2015)
2. Wolf, M.: The Medium of the Video Game Republished edn. University of Texas Press (2001)
3. Palazzi, C., Roccetti, M., Marfia, G.: Realizing the Unexploited Potential of Games on Serious Challenges. Comput. Entertain. 8(4), 1-4 (December 2010) doi:10.1145/1921141.1921143.
4. Meyer, A., Zimmermann, T., Fritz, T.: Characterizing Software Developers by Perceptions of Productivity. In : 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), pp.105-110 (2017) doi:10.1109/ESEM.2017.17.
5. Haas, J.: A History of the Unity Game Engine. Worcester Polytechnic Institute, Worcester, USA (March 2014)
6. Packer, B., Keates, S., Baker, G.: Game Developers in the Wild: Trending Perspectives on Software Limitations. Multimed Tools Appl (2021) Publication Pending.
7. Gibson, W., Brown, A.: Working with Qualitative Data. SAGE, Los Angeles, USA (2009)
8. Renz, S., Carrington, J., Badger, T.: Two Strategies for Qualitative Content Analysis: An Intramethod Approach to Triangulation. Qualitative Health Research 28(5), 824-831 (2018) doi:10.1177/1049732317753586.
9. Wilson, C.: Chapter 2 - Semi-Structured Interviews. In : Interview Techniques for UX Practitioners: A User-Centered Design Method. Morgan Kaufmann, Boston (2013) 23-41 doi:10.1016/B978-0-12-410393-1.00002-8.
10. Pribeanu, C., Balog, A., Iordache, D.: Measuring the perceived quality of an AR-based learning application: a multidimensional model. Interactive Learning Environments 25(4), 482-495 (2014) doi:10.1080/10494820.2016.1143375.
11. Hariyanto, D., Triyono, M., Koehler, T.: Usability evaluation of personalized adaptive e-learning system using USE questionnaire. Knowledge Management and E-Learning 12, 85-105 (March 2020) doi:10.34105/j.kmel.2020.12.005.
12. Kress, G., Hoster, H., Chung, C., Steinert, M.: Headspace: The Stanford imaginarium. In : ICDC 2012 - 2nd International Conference on Design Creativity, Proceedings, Glasgow, vol. 2, pp.261-268 (2012)
13. Ali-Babar, M.: The Application of Knowledge-Sharing Workspace Paradigm for Software Architecture Processes. In : SHARK '08: Proceedings of the 3rd international workshop on Sharing and reusing architectural knowledge, Leipzig, pp.45-48 (2008) doi:10.1145/1370062.1370074.
14. Rowley, J.: Using Case Studies in Research. Management Research News 25(1) (January 2002) doi:10.1108/01409170210782990.
15. Weston, C., Gandell, T., Beauchamp, J., McAlpine, L., Wiseman, C., Beauchamp, C.: Analyzing Interview Data: The Development and Evolution of a Coding System. Qualitative Sociology, Vol. 24, No. 3, 2001 24(3), 381–400 (2001) doi:10.1023/A:1010690908200.

16. Dave, K., Lawrence, S., Pennock, D.: Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. In : WWW '03: Proc. 12th Int. Conf. World Wide Web, Budapest, pp.519–528 (2003) doi:10.1145/775152.775226.
17. Suddaby, R.: From the Editors: What Grounded Theory Is Not. The Academy of Management Journal 49(4), 633-642 (August 2006)
18. Leroux, J., Rizzo, J., Sickles, R.: The Role of Self-Reporting Bias in Health, Mental Health and Labor Force Participation: A Descriptive Analysis. Empirical Economics 43, 525–536 (2012) doi:10.1007/s00181-010-0434-z.
19. Flyvberg, B.: Five Misunderstandings About Case-Study Research. Qualitative Inquiry 12(4), 219-245 (April 2006) doi:10.1177/1077800405284363.