Codeless3D: Design and Usability Evaluation of a Low-Code Tool for 3D Game Generation

Christina Volioti, Vasileios Martsis, Apostolos Ampatzoglou, Euclid Keramopoulos, Alexander Chatzigeorgiou

***Abstract*—In recent years, the game industry has experienced significant growth from both a financial and a social viewpoint. Developing compelling games that rely on novel content is a challenge for 3D firms, especially in terms of meeting the diverse expectations of end users. Game development is performed by multidisciplinary teams of professionals, in which game / level designers play a crucial role. Inevitably, they often depend on programmers for technical implementations creating bottlenecks, even for prototyping purposes. This issue has raised the need for introducing efficient low-code environments that empower individuals without programming expertise to develop 3D games. This work introduces Codeless3D, a prototype for low-code 3D game creation by non-programmers. The proposed approach and the tool aim to reduce design and development time, bridging the gap between conceptualization and production. To evaluate the usefulness of Codeless3D, in terms of usability and its vision, an evaluation study was conducted. The findings suggested that Codeless3D effectively reduces design and development time for stakeholders in the game development field. Overall, this paper contributes to the emerging trend of low-code tools in the entertainment domain and offers insights for further improvements in game design and development processes.**

***Index Terms*—** Game Development, Game Design, Game Design Document, Low-Code, Usability

I. INTRODUCTION

OVER the last decade, the game industry has grown exponentially, exhibiting a revenue growth estimated to reach 285bn dollars in 2027[1]. The impact of the game industry extends beyond financial gains, in the sense that it also has a profound social effect. Games have become a prominent form of entertainment, fostering social interaction, and providing immersive experiences. Games target quite diverse groups, making the expectations of the end users hard to predict, and even harder to meet. Therefore, game firms struggle to develop compelling new games and create novel content [1] that will safeguard their position in the market. Additionally, games are an extremely complex product to develop, therefore the end-to-end development process (*shaping the game idea into a product*) is far from trivial [2]. Furthermore, throughout the iterative process of game development, a diverse range of professionals (such as programmers, animation programmers, artists, game / level designers, sound engineers, testers, etc. [3][4]) actively participate and collaborate closely to envision, design, and implement a comprehensive game product. This implies that the game development team does not only comprise programmers, but also a multitude of non-programmers, who specialize in the creative / artistic and conceptual aspects of game creation, cumulatively referred to as the game design team. While it is possible for some team members to possess programming skills and contribute to code implementation, such cases are not the norm, resulting in a significant reliance on the programmers for technical implementation, even for the creation of prototypes. Consequently, an imperative necessity has been raised for low-code environments that cater to the specific needs of non-programmers, such as *game / level designers*, facilitating their ability *to produce small to medium-sized and -complexity games and 3D experiences* (***need #1***). These environments will aim to empower individuals without programming expertise, enabling them to actively participate and contribute to the creation of games.

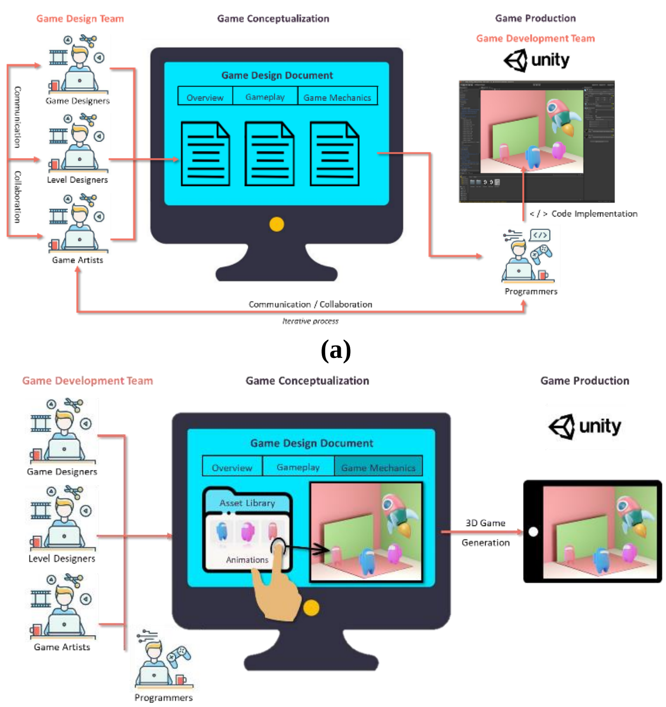
Figure 1(a) visualizes the current state of the game development process, where the first phase is the *game conceptualization*, a challenging and intensive process [5] in which all members of the game development team must effectively communicate, collaborate, and comprehend the game concept. This phase sets the foundation for game design and shapes the overall vision and direction of the complete game development process. During the conceptualization phase, various aspects need to be considered and defined: for instance, according to Baldwin [6] the game design team needs (at minimum) to define the ***game overview*** (e.g., core concept, genre, target audience, scope), the ***gameplay*** (e.g., game progression, objectives), and the ***mechanics*** (e.g., rules, physics, actions, combat). In the game development industry, the key artifact for documenting these aspects is the Game Design Document (GDD), which despite its various formats and level-of-detail is developed for most game projects [7]. The GDD, apart from specification purposes, serves to facilitate the exchange of ideas and acts as the blueprint for the final product [3]. By investing time and effort in the conceptualization stage, the game design team can establish a solid foundation for the game development. It is worth noting that although the GDD can be considered as a living document that undergoes evolution and iteration during game development [8], it primarily

This research was supported by Action for Promoting Research Production *(Corresponding author: C. Volioti).*
V. Martsis, and E. Keramopoulos are with the Department of Information and Electronic Engineering, International Hellenic University, P.O. Box 141, 574 00 Sindos, Greece, (e-mail: vasilismartsis@yahoo.gr, euclid@ihu.gr).
C. Volioti, A. Ampatzoglou, A. Chatzigeorgiou are with the Department of Applied Informatics, University of Macedonia, 156 Egnatia, Thessaloniki, Greece (e-mail: chvolioti@gmail.com, apostolos.ampatzoglou@gmail.com, achat@uom.edu.gr).

[1]https://www.statista.com/forecasts/456595/video-games-revenue-in-the-world-forecast

comprises static textual information. It does have the potential to incorporate visual elements like diagrams and mock-ups, which can enhance comprehension and communication, however, this necessitates the use of supplementary digital tools that require technical knowledge (***need #2***).

Subsequently, in the *game production* phase (Fig.1(a)), the development team uses GDD as a reference and ***transforms it into actual source code*** [3]. The production phase is iterative and involves continuous testing, debugging, and refinement of the source code. Programmers collaborate with other team members, (e.g., artists, designers), to ensure that the implemented code aligns with the envisioned gameplay and visual experience. The main challenge here is whether and to what extent the actual game reflects the GDD, since its translation (e.g., game mechanics) to functional requirements [9] is not always an easy task. Since prototypes are constructed after the main idea has been outlined in the GDD to assess the viability of game mechanics, demonstrate ideas, and test technical aspects, in many cases there is a gap between the conceptualization and the implementation of the game [10] (***need #3***).



(a)

(b)

Fig. 1. (a) Current State of the Game Development Process, (b) Overview of Envisioned Approach.

Considering these needs, the overview of the envisioned approach is illustrated in Figure 1(b), aiming to introduce a low-code design tool, which will allow all members of the development team, regardless of their coding expertise, to generate 3D games, reducing the iterative process as well as speed up the overall design and development timeline. To this end, we propose Codeless3D, which is a prototype for 3D game generation. We should stress that only a limited number of features of the envisioned approach are implemented in the current version. To empirically assess the proposed approach, an evaluation study was conducted to investigate its usability (effectiveness, efficiency, and level of user satisfaction) and assess our vision (industrial relevance, readiness, and acceptance). Our key contributions are summarized as follows: (a) introducing Codeless3D, a prototype tool for creating 3D games without requiring programming expertise; (b) conducting an evaluation study for assessing the usability and vision of Codeless3D; and (c) providing insights for future improvements and implications for researchers and practitioners.

II. Background Information and Related Work

Salen and Zimmerman [11] defined a game as "*a system in which players engage in an artificial conflict, defined by rules, that results in a quantifiable outcome*" and is classified into a wide range of genres, such as shooter games, role-playing games, sports games, adventure games, etc. [12]. The development of such a game inherently depends on the creative skills of the designers [13]. For the designers to express their concept to the programmers the GDD is produced, which is the main artifact of the game design process. However, designers might not have the appropriate writing skills to produce such a document [14], or may make revisions to the GDD that might be time-consuming and unproductive [5], resulting in a complex, huge, and hard-to-understand GDD for the development team. Therefore, the size and the format of the GDD are factors that need to be highly considered [13], otherwise the following dilemma arises. Although some game designers believe that too much structured GDD at the beginning of game development might have negative consequences such as stifling creativity and limiting expression [15], the majority highlights the importance of a formal structure that can result in a good-quality game as the development team can quickly move into the game production [13][14][16].

Towards the direction of GDD formalization, Atmaja and Parlika [2] presented a preliminary study of implementing a formal structure by applying the Mechanics-Dynamics-Aesthetics (MDA) framework [17][18] into the GDD, to generate game level procedurally. Levels along with maps, textures, quests, music, characters, game rules, etc., are part of the game content [19]. Thus, the term Procedural Content Generation (PCG) is used to refer to the automatic creation of game content that can be generated either on its own or by a human using algorithms [20]. Another research work on the automation and formalization of game design is ANGELINA system [21][22], which automatically generates 3D simple games by using AI taking into account thematic elements as well as mechanics of the game's design. Dormans [23] proposed Machinations, a formalized design tool focused on modeling game mechanics to promote the use of Model Driven Engineering in game design. Similarly, Schaul [24] and Perez-Liebana et al. [25] developed the Video Game Description Language in Python and Java respectively, to describe a wide range of 2D games with visual logic.

In addition to formalization, several approaches operationalize the concept of micro-rhetoric. Such an approach is the tool Game-o-Matic introduced by Treanor et al. [26], where a concept map is used with simple relationships as an input that constitutes the set of rules to automatically generate simple arcade-style games. Summerville et al. [27] expanded upon this research with Gemini, by making the game generation bi-directional and capable of both interpreting a specification to generate a game, and interpreting a game to generate a specification. Going one step further, Kreminski et al. [28] developed Germinate, an extension of Gemini that is an open-source casual creator for rhetorical game design. Casual creators refer to tools that prioritize the creativity over quality and target non-programmers [29], who are curious to explore [30].

Finally, there are some studies that focus on recreating existing games. For example, Guzdial and Riedl [31][32] introduced the technique of conceptual expansion, in which characteristics of different existing games were combined to automatically produce a game, and evaluated their approach with three games from the Nintendo Entertainment System (Super Mario Bros, Kirby's Adventure, and Mega Man) as input. Other examples are Baba is Y'all [33], Anhinga (clone of Snakebird) [34] and Ropossum (clone of Cut The Rope) [35], which are mix-initiative design tools that allow users to regenerate existing games.

Summarizing, the above research works investigated the formalization of the game design combined with the PCG. It is worth noting that most of these studies focused on generating 2D games rather than 3D games. In addition, while PCG offers a variety of benefits like infinite possibilities and replay value [36][37], it also brings potential disadvantages. One of the main drawbacks is the lack of direct control over specific details and handcrafted content, which may result in a loss of game designer intent and a decrease in overall coherence and consistency [38]. This is where the strengths of low-code tools can become evident. These tools, which represent an emerging trend, provide graphical user interfaces that simplify the design process by offering drag-and-drop functionality [39]. By leveraging such tools, users can focus more on the creative aspects of design [39], such as aesthetics and experience creation, rather than the technical details. Moreover, they can ensure that the generated content is aligned with the intended gameplay experience, maintaining coherence, and delivering an engaging game world. It is important to note that the terms "*low-code*" and "*no-code*" are often used interchangeably [39], even though there is a subtle distinction implied by their names, suggesting that low-code tools involve minimal reliance on textual programming languages compared to no-code solutions. Additionally, low-code tools target both non-programmers and professional programmers [40][41], offering a middle ground between traditional coding and visual development. While no-code / low-code tools are commonly employed in domains like e-commerce and business project management, their application in the entertainment domain is limited [39]. Additionally, the majority of these tools are commercially available (e.g., XR+[2], Zapworks[3], 8th Wall[4], Vossle[5], etc.) with only a few research papers published on the subject. As an example, consider the research work conducted in the domain of business project management, where the authors developed a no-code authoring tool called WizARd [42]. This tool aims to assist users in creating business process guidance systems and providing on-site assistance by leveraging AR. In the healthcare field, a low-code VR authoring platform called MAGES SDK was introduced [43], which enables the rapid creation of high-fidelity collaborative medical training simulations in virtual reality and augmented reality. Another noteworthy contribution by Fleck et al. [44] involved the development of a versatile low-code toolkit for situated analytics. This toolkit offers the advantage of being a general-purpose toolkit capable of building various successful application prototypes. Moreover, Torres et al. [45] proposed a no-code virtual serious game authoring platform specifically designed for nursing educators. This platform empowers educators to design serious games that focus on the development of decision-making and communication skills. Finally, there are some low-code platforms, such as Scratch[6], Unreal Blueprints[7], GameMaker[8] and PlayMaker[9] for Unity3D, that focus on game development. However, users may encounter challenges when trying to implement advanced game mechanics, because the tools provide limited guidance, and therefore programming knowledge may be required to overcome these limitations and create more complex games.

The aforementioned approaches are not an exhaustive literature review; rather, they serve as indicative examples to emphasize the necessity of a low-code design tool in the domain of game development that seeks to achieve the following objectives: (a) empower all the members of the game development team, regardless of their programming expertise, to generate a game; (b) generate a small-sized and -complexity 3D platform game; (c) support the formalization of the game design by providing a structured way for specifying the characteristics of the game; and (d) reduce both the design and the development time required for creating a game.

III. Codeless 3D Overview

This section presents the description of the proposed approach. It is important to note that trying to integrate existing solutions would have been a viable alternative to developing the system from scratch (as opted for in this work). However, using existing solutions poses a non-negligible risk of failure at integration stage. Initial requirements on the envisioned approach have been gathered from three Game Development companies in Greece; so, the target is to develop a novel tool tailored to their needs. Thus, step-by-step development, evaluation, and feedback iterations were preferred in this direction, to allow incremental development. As a result, the objective of this tool is to facilitate both programmers and game / level

[2] https://xr.plus/
[3] https://zap.works/
[4] https://www.8thwall.com/
[5] https://vossle.ai/
[6] https://scratch.mit.edu/
[7] https://www.unrealengine.com/en-US
[8] https://gamemaker.io/en
[9] https://assetstore.unity.com/packages/tools/visual-scripting/playmaker-368

designers to generate small-sized and -complexity 3D platform games (a sub-genre of action video games) primarily focused on collision logics, where virtual objects can be touched triggering possible events [46]. Codeless3D is designed to simplify the design and development process, enabling all members of the development team to create a 3D game effortlessly. Codeless3D[10] consists of two main components: (a) the ***Scene Initializer***, and (b) the ***Scene Importer***. The Scene Initializer (Design Phase) serves as a way for the user to enter the basic characteristics and specifications for the game. Once the necessary information has been filled in, the Scene Initializer exports the data into a JSON file format. This JSON file can then be imported into the Unity3D Game Engine through the component called Scene Importer (Development Phase). Unity3D serves as the development environment for the generation of the actual 3D game. It is important to note that the current version of Codeless3D is a prototype. Therefore, as part of the initial demonstration and validation of the concept, limited functionalities have been implemented. Figure 2 illustrates the high-level architecture of the prototype Codeless3D, while subsequent subsections provide a more comprehensive breakdown of the phases involved.

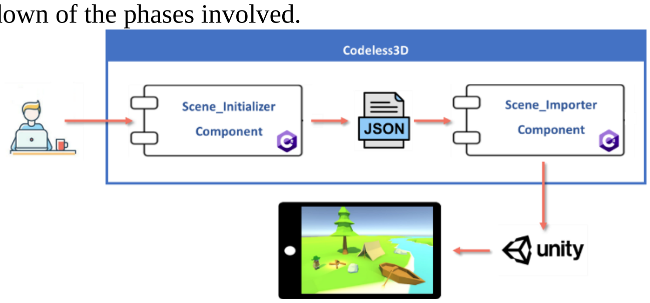


Fig. 2. Overall Architecture of Codeless3D.

A. Design Phase

During the Design phase, specific information about key game mechanics (e.g., player movement, collision, physics, etc.) [47][48] and game elements (e.g., materials, textures, etc.) is gathered for the development of a 3D game within Unity3D. To achieve this, a low-code interface is provided, enabling the user to create objects (known as `GameObject`) within the game. The user undergoes an iterative process using the UI, creating multiple game objects, and assigning different attributes to each one. This allows the user to define and customize the properties of each object. This component is built with the C# and the .NET Framework. Figure 3 provides an overview of the UI used in this phase. The user interface prompts the user to assign a Name to each `GameObject`, which serves as a general description. Next, the user has the option to upload the desired model, typically in FBX format, along with its corresponding `Textures`, `Materials`, and `Animations`. In the current version of the tool, the user adds `Animations {True, False}` and optionally selects up to four different animations: `Idle`, `Walking`, `Running`, and `Jumping`. These animations are typically ANIM files, and each animation corresponds to one of the mentioned states (Idle, Walking, Running, Jumping).

The next group of attributes concerns the physical aspects of the `GameObject`, by specifying the `Position`, `Rotation`, and `Scale` of the object. These attributes define the object's location, orientation, and size within the game environment. Next, the user selects the `Collider`, which defines the object's shape for the purposes of collision detection. Additionally, the user sets up the `Physics {True, False}` of the object by defining rules such as gravity. These physics rules govern how the object interacts with other objects and the environment. Moving on to user interaction, there are two attributes to consider. The first one allows the user to choose whether s/he will control the movement of the `GameObject` in the scene. The second attribute concerns the user's view of the scene. The user decides whether the current `GameObject` will carry a camera and selects the type of camera, such as a `First-Person` or a `Third-Person` camera. Furthermore, the user can specify the amount of `Health` the `GameObject` will have. As well as s/he can determine whether the `GameObject` can take damage or not, which affects its resilience in the game.

Fig. 3. UI of the Design Phase (Scene Initializer component).

The above steps are repeated until all the `GameObjects` in the scene and their attributes are defined. This streamlined process ensures that the required elements are accurately con-

[10] https://github.com/game-dev-kit/Codeless3D

nected to each `GameObject`, setting the stage for the next steps of the 3D game generation. Therefore, once the Design Phase is complete, the user initiates the game export process by clicking the "Create Your Unity Package" button. This process generates a Unity3D package that contains the information linked to mechanics and game elements in a JSON format. It is worth noting that it can be utilized for creating both 2D and 3D games, as shown in Figure 3 (`Dimension`).

B. Development Phase

In the Development phase, game generation occurs within the Unity3D Game Engine using a C# component. To begin the process, the user needs to install the generated Unity3D package into a Unity Project. Once installed, an additional menu option called "Import Scene" becomes available in the Unity3D UI. By selecting this option, the generation of the 3D game starts based on the information stored in the JSON file. It is worth mentioning that in the current version some specific steps are not explicitly selected by the user in the Design Phase but are implemented at the beginning of the generation process. Such steps are the pre-defined illumination and the terrain that are automatically added to the scene.

After these pre-defined steps, the instantiation of each `GameObject's` model occurs during the game generation, and specifically, the models specified in the JSON file are automatically added to the scene along with their textures, materials, and animations. To handle animations, an Animator Component of the `GameObject` is created and given the four default empty animations. Based on the JSON file, the appropriate animation is assigned to each model. Additionally, the desired `Position`, `Rotation`, and `Scale` values are assigned to each `GameObject`. The instantiation of the camera follows, offering three options: a `Static`, a `First-Person`, and a `Third-Person` camera. Based on the JSON file, the user's chosen camera type during the Design Phase is implemented. In the current version, only one object in the scene can carry the camera, and it is the object specified as controllable in the JSON file. Physics is also added to the selected `GameObjects` by attaching a Rigid Body Component to each one. This allows the models to be affected by gravity and interact with other models in the scene. Interactions are achieved by adding a collider to the `GameObject`, based on the JSON file data. The player's interaction with the 3D models is facilitated through the model designated as controllable in the JSON file. Finally, each `GameObject` is assigned health attributes and the ability to receive damage or not, as specified in JSON. To better illustrate the aforementioned processes, we have created and published a walkthrough on using Codeless3D to develop a sample scene[11].

IV. Evaluation Study Design

An evaluation study was conducted to assess the *usability* and the *vision* of the proposed approach and prototype tool. The study design adheres to the guidelines of Runeson et al. [49].

A. Objectives and Research Questions

The evaluation is based on the Goal-Question-Metric (GQM) approach [50] and initially aims to evaluate the *usability* of Codeless3D from the perspective of the game programmers and game/level designers. In accordance with ISO 9241-11, usability is evaluated based on the following metrics [51]:

Effectiveness refers to the accuracy and completeness with which users can perform tasks using Codeless3D. It focuses on the extent to which users can achieve their goals and successfully accomplish the required actions.

Efficiency measures the time taken by users to complete specific tasks using Codeless3D. It assesses the speed and productivity of users in performing their tasks, aiming to identify any potential bottlenecks or areas where improvements can be made to optimize the workflow.

Satisfaction evaluates users' overall impression and experience with Codeless3D. It encompasses users' subjective feelings, perceptions, and opinions about the tool's usability, ease of use, and overall enjoyment of the interaction. This metric provides insights into user preferences, comfort levels, and potential areas of improvement for enhancing user satisfaction.

Secondly, it aims to investigate the *vision* of Codeless3D by assessing: (a) the **industrial relevance** of the envisioned approach; (b) the **readiness** of the existing prototype tool; and (c) the pathway to **industrial acceptance**.

Therefore, based on the goals of the study, the following research questions have been set:

RQ₁ What is the effectiveness of Codeless3D?
RQ₂ What is the efficiency of Codeless3D?
RQ₃ What is the satisfaction of using Codeless3D?
RQ₄ What is the industrial relevance, readiness, and acceptance of Codeless3D?

B. Case and Task Selection

To address the research questions, an exploratory evaluation study was conducted involving twenty (n=20) stakeholders, out of whom 12 were males and 8 were females, with an average age of 32 years old. They were all professionals from the game industry with varying levels of experience in game programming, and specifically in using Unity3D (Figure 4). In the context of our study, non-programmers were mainly game and level designers, who had no prior programming experience with Unity3D, however, they may have had exposure to scripting languages within their companies.

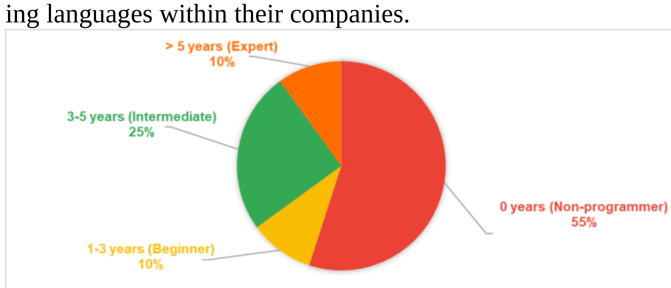


Fig. 4. Level of Experience in Game Programming.

The evaluation study was organized as a half-day workshop in Greece. Participation was entirely voluntary, with participants providing consent, and all data gathered during the study

[11] https://game-dev-kit.github.io/toolkits/codeless3d/

was treated as anonymous and confidential. In Appendix A, we present the participant information sheet. The study began with a concise introduction that outlined the research goal and the problem under investigation. The workshop consisted of four distinct phases, designed to systematically gather data on the usability and vision of Codeless3D, which are as follows:

Phase 1: Introduction and Task Discussion
The researchers introduced the concept of the workshop to the participants (20 minutes). Next, participants were provided with a set of tasks and were given 10 minutes to understand the tasks and discuss any questions they had. This phase ensured clarity and comprehension among the participants before proceeding to the actual evaluation study.

Phase 2: Evaluation Study Task Completion
Participants were given 40 minutes to complete the assigned tasks using Codeless3D (the tasks are presented below). During this phase, participants had the opportunity to interact with the tool and evaluate its effectiveness in accomplishing the given tasks. We note that the tasks have been completed without access to tutorials or demonstrations from the researchers. Therefore, they correspond to the worst-case scenario.

Phase 3: Usability Questionnaire
After completing the tasks, participants were given a usability questionnaire to assess their satisfaction and overall experience with Codeless3D. They were allotted 10 minutes to complete the questionnaire, which contained a range of relevant questions pertaining to usability metrics. The questionnaire was extracted from the literature and is considered as state-of-the-art in the domain of usability.

Phase 4: Focus Group
A 60-minute discussion was conducted, focusing on specific questions related to Codeless3D usability and vision. Participants were encouraged to share their insights, provide feedback, and engage in a detailed analysis of their experience using the tool. This phase aimed to capture qualitative data and gather their subjective opinions and impressions.

Next, we focus on the tasks that were used to assess Codeless3D. The tasks were designed to cover various aspects of the tool's functionality and evaluate its effectiveness in facilitating the game development process. Participants were expected to complete these tasks using Codeless3D within the designated time frame. The activity description and the tasks that were given to participants to generate the 3D game are presented below.

Activity Description: The users were asked to generate a small-sized and -complexity 3D platform game that focuses on collision logics, using Codeless3D. The 3D game consisted of one scene (one level) that included the default terrain and illumination, and users had to add the seven following 3D models: (1) human, (2) fire logs, (3) tree, (4) boat, (5) tent, (6) big rock, and (7) small rock; adding different properties to each one. The resulting 3D game is expected to resemble Figure 5.

Fig. 5. The generated 3D game from Codeless3D.

Design Phase:
T1. Run the "UnityPackageGenerator"
T2. Define the scene as 3D
T3. Give the 3D object a name
T4. Select one of the seven models with its corresponding textures, materials, and animations to upload
T5. Add coordinates (x, y, z) to the object
T6. Add rotation (x, y, z) to the object
T7. Add size (x, y, z) to the object
T8. Select a collider for the object
T9. Select if the object has physics
T10. Select if the object has gravity
T11. Select if the object is controllable
T12. Select if the object has a camera
T13. Select the type of camera
T14. Add health to the object
T15. Repeat T4-T14, to include the rest six 3D models and select different properties for each model
T16. Click "Create Your Unity Package" and select the location where the Unity Package will be saved

Development Phase:
T17. Open Unity3D Game Engine
T18. Select the "package.json" file and add it to the Unity3D
T19. The Unity Package has been integrated and the new menu called "Json Scene Generator" is appeared, click it and then click "Import Scene", with which the 3D game is generated
T20. Press "Play" to play the 3D game with the seven different 3D models and the corresponding animations

C. Data Collection

To achieve data triangulation, we relied on various data collection methods—see Table I. The first data collection method, **Task Analysis**, was used to gather data for answering RQ_1 (effectiveness) and RQ_2 (efficiency). This method involved analyzing the participants' performance while completing specific tasks, recording any errors or difficulties encountered, and measuring the time taken to complete each task.

TABLE I: DATA COLLECTION METHODS PER RQ

Collection Method	RQ_1	RQ_2	RQ_3	RQ_4
Task Analysis	X	X		
Questionnaire			X	
Focus Group	X	X	X	X

The second data collection method, **Questionnaire**, was used to evaluate the level of satisfaction obtained by using

Codeless3D (RQ_3). The participants were provided with a 5-point Likert scale questionnaire, specifically the System Usability Scale (SUS) [52] (see Table II), to assess their satisfaction and overall impression of Codeless3D. One notable benefit of the SUS is its remarkable effectiveness in terms of both reliability [53] and validity [54]. Moreover, the SUS demonstrates its reliability by producing consistent and dependable results, even when working with a limited sample size [54]. Finally, we have conducted a **Focus Group** to gather qualitative insights and feedback from the participants regarding all research questions. The focus group discussion allowed an in-depth exploration of participants' experiences, perceptions, and suggestions related to Codeless3D.

TABLE II: SYSTEM USABILITY SCALE

#	Question
1	I think that I would like to use this system frequently
2	I found the system unnecessarily complex
3	I thought the system was easy to use
4	I think that I would need the support of a technical person to be able to use this system
5	I found the various functions well integrated
6	I thought there was too much inconsistency
7	I would imagine that most people would learn to use this system very quickly
8	I found the system very cumbersome to use
9	I felt very confident using the system
10	I needed to learn a lot of things before I could get going with this system

D. Data Analysis

To evaluate Codeless3D both quantitative and qualitative analysis was employed. The quantitative analysis focused on measuring the usability and vision of Codeless3D. For RQ_1 (effectiveness), the overall effectiveness of Codeless3D was measured by calculating the average percentage of correctly executed tasks. This provided an indication of how accurately and successfully the participants were able to perform the assigned tasks (T1 - T20) to using the tool. Regarding RQ_2 (efficiency), the following metrics were considered:

- the ***average completion time*** for each task was recorded to assess the time efficiency of using Codeless3D;
- the ***number of errors*** made by the participants during task completion was considered; and
- the ***success or failure of each task*** was also considered as an efficiency measure.

For RQ_3 (user satisfaction), the SUS questionnaire was utilized. The questionnaire consisted of 10 statements that participants responded to using a 5-point Likert scale, ranging from "Strongly Disagree" to "Strongly Agree". Each statement was assigned a score, with some statements receiving reverse scoring. The scores for all ten statements were then summed to obtain a total score between 0 and 100. Higher scores indicated higher perceived usability, with scores above 68 considered average usability [52].

Additionally, to obtain qualitative results, the data gathered from the focus group were used, applying the Qualitative Content Analysis (QCA) technique [55] that is for the subjective interpretation of the content of text data. The process included data preparation, where the text was organized and made ready for analysis. Then open coding took place by assigning codes (i.e., words, phrases, short descriptions) to meaningful units of text. These codes were grouped together to form categories, and through an iterative process, higher-level themes or categories were developed. The data were abstracted and summarized within each category to capture its essence. The interpretation of the findings occurred by analyzing relationships, patterns, and meanings in the data. Finally, the results were reported through narrative descriptions, and direct quotes, providing a comprehensive understanding of the analyzed content.

V. RESULTS

The findings of the analysis are presented in this section, and organized according to each research question. Regarding the qualitative analysis, codes are presented in capital letters and quotes in italics. Table III presents the codes that emerged from the focus group discussions as well as how many participants mentioned each one.

TABLE III: CODES OF THE QUALITATIVE ANALYSIS

Code	No. of Participants
COMPREHENSIVE GAME CREATION	17
STREAMLINED TASK COMPLETION	20
TIME SAVING	20
DIRECT IMPLEMENTATION	20
USER-FRIENDLY INTERFACE	12
ALL-IN-ONE TOOL EXPERIENCE	16
SCALABILITY	12
FEATURE COMPLETENESS	12
CONTINUOUS IMPROVEMENT	18

RQ_1 (Effectiveness): Table IV presents the completion rates for each task. All tasks were successfully completed by the majority of participants, except for tasks T14, and T17 - T20. Specifically, T14 was not completed by one participant unintentionally, as he was engrossed in the activity and inadvertently skipped it while proceeding to the next tasks. As for T17 - T20, three participants were unable to complete them within the allocated 30-minute timeframe due to the extended loading time of Unity3D on their laptops. Additionally, the overall effectiveness of Codeless3D was determined to be 97% indicating that the tasks were generally perceived as easy to comprehend and perform.

Taking also into consideration the discussion that was held in the focus group, 17 out of 20 participants (85%) identified COMPREHENSIVE GAME CREATION as an advantage. They emphasized their ability to actively engage and successfully generate a new 3D game from scratch, irrespective of their prior experience in 3D game development. This is proven by their enthusiastic statements *"…it has transformed how I approach game design…"* (P13), *"…I didn't feel overwhelmed or encounter significant problems because the creation process was smooth…"* (P15), and *"…previously, I jumped between various tools for my game ideas, but now, I can manage the entire lifecycle of game development through this tool…"* (P20). In addition, all 20 participants (100%) concluded that *"…the delays that occurred were largely attributed to external circumstances rather than the inherent difficulty of the tasks…"*, further reinforcing the benefit of `STREAMLINED TASK COMPLETION`.

TABLE IV: COMPLETION RATES PER TASK

Task No.	Completion Rate	Task No.	Completion Rate	Task No.	Completion Rate	Task No.	Completion Rate
T1	100%	T6	100%	T11	100%	T16	100%
T2	100%	T7	100%	T12	100%	T17	90%
T3	100%	T8	100%	T13	100%	T18	90%
T4	100%	T9	100%	T14	95%	T19	85%
T5	100%	T10	100%	T15	100%	T20	85%

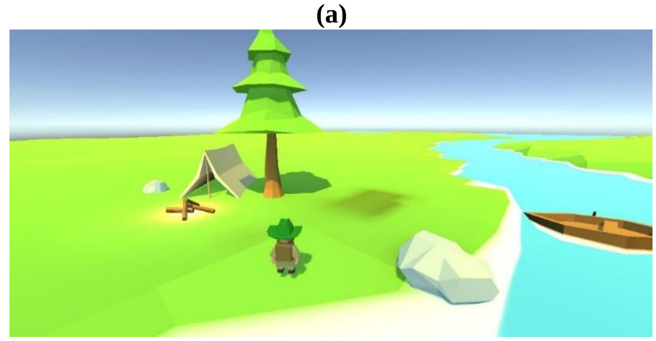
RQ$_2$* *(Efficiency): Table V presents the task completion rates, errors, and average task completion times for each task (in minutes). Except for T14, which one participant accidentally did not complete due to eagerness to progress, no errors were observed, since T17 - T20 were not completed at all. The average time taken to complete the entire activity was 25.4 minutes, which is within the allotted maximum time of 30 minutes. Furthermore, the average task completion time, excluding T15, was 0.5 minutes, indicating efficient task execution. It is worth mentioning that the time taken to complete T15 was 15.9 minutes, as participants had to repeat tasks T3 - T14 an additional six times. Each repetition took an average of 2.7 minutes, which is less than the initial completion time of 4.5 minutes for T3 - T14. Thus, the task completion time exhibited a significant improvement of 40%, indicating that both the design as well as the development time for creating a new game can be significantly reduced.

The efficiency of Codeless3D aligns with the feedback provided by the focus group. All 20 participants (100%) emphasized that Codeless3D is `TIME SAVING` and supports `DIRECT IMPLEMENTATION`. Specifically, they expressed that the tool significantly reduced both the design and development time as well as the effort required to create an entirely new 3D game (*"…I went from concept to execution in record time…"* (P5)). The positive feedback from participants further reinforces that Codeless3D effectively streamlines the game development process, allowing users to achieve their goals more efficiently and with reduced errors through an intuitive and low-code interface with tasks that are easily comprehensible and executable.

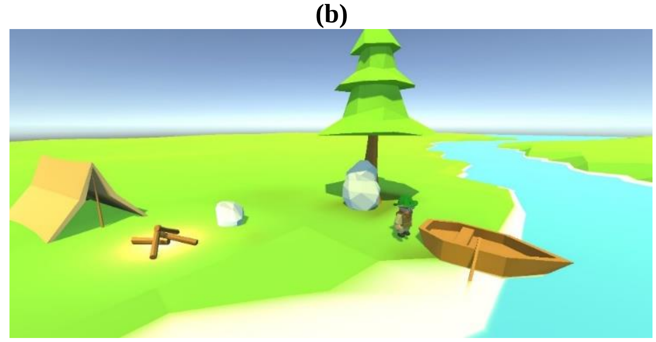
TABLE V: SUMMARY OF TASK COMPLETION

Task No.	Task Completion	Errors	Average Time on Task (in minutes)	Task No.	Task Completion	Errors	Average Time on Task (in minutes)
T1	20	0	0.5′	T11	20	0	0.3′
T2	20	0	0.2′	T12	20	0	0.3′
T3	20	0	0.3′	T13	20	0	0.3′
T4	20	0	1.1′	T14	19	1	0.3′
T5	20	0	0.3′	T15	20	0	15.9′
T6	20	0	0.4′	T16	20	0	0.9′
T7	20	0	0.3′	T17	18	0	1.7′
T8	20	0	0.4′	T18	18	0	1′
T9	20	0	0.3′	T19	17	0	0.5′
T10	20	0	0.2′	T20	17	0	0.2′

(a)



(b)



(c)

Fig. 6. The result of the 3D game of P2, P9, and P13.

In addition, Figure 6 presents the outputs of the complete 3D games created by participants P2, P9, and P13. The generated scenes included seven distinct 3D models, each assigned with different properties such as position, rotation, scale, and more. The terrain and illumination settings were set to their default properties.

RQ3 (User Satisfaction): Codeless3D achieved a total SUS score of 79, which falls within the acceptable range, "good" for adjective and of usability grade B [53]. Figure 7 displays the scores for individual questions, revealing that "Background Knowledge", "User Confidence", and "Integration" received lower satisfaction ratings. To interpret these findings, the results presented in Figure 8, which illustrate the SUS per participant as well as the results of the discussion in the focus group, should be considered and correlated.

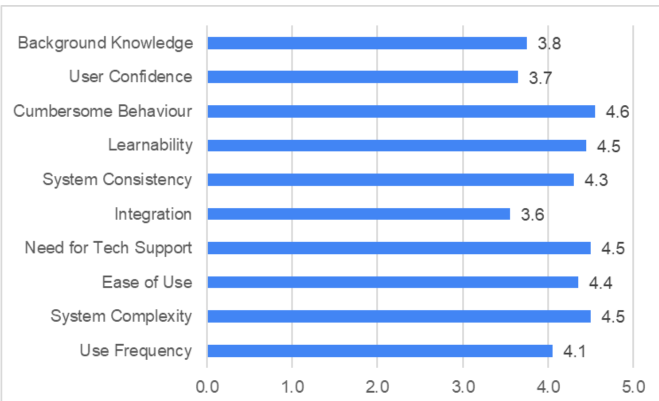


Fig. 7. SUS Score per SUS Question.

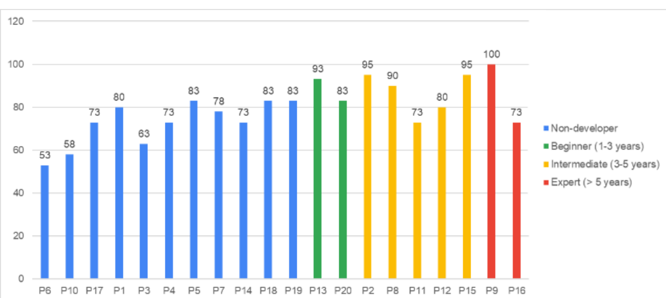


Fig. 8. SUS Score per Participant.

Among the game / level designers, and especially P6 and P10, it was observed that they initially found it quite difficult to grasp the terminology and required some assistance when they used Codeless3D for the first time. Although Codeless3D offers tooltips for object attributes, they found certain explanations to be vague. Consequently, their confidence in using Codeless3D was affected, and they expressed a need for more comprehensive explanations.

This is consistent with the results of the focus group, where only 12 participants (60%) mentioned the benefit of USER-FRIENDLY INTERFACE, supporting that although the interface of Codeless3D *"...is very simple and easy to use, the design needs to be improved..."* (P11). They mentioned a better organization of the sections, especially in the case of adding more GameObjects as *"...the scroll down even for the seven models is quite complex..."* (P17). In addition, although they agreed that better explanations are needed, they acknowledged that *"...the repetition of tasks significantly reduced the time required to execute them..."* (P13), thus enhancing their overall learning experience with the tool. Finally, in terms of ALL-IN-ONE TOOL EXPERIENCE, 16 participants were positive (60%), while the rest 4 (and specifically P6, P10, P3, and P16) suggested that the two separate components of Codeless3D should be integrated into a single and unified component within Unity3D because this would make the whole process even easier.

RQ4 (Vision): Regarding the vision of Codeless3D, the main argument that 12 participants (60%) have used to champion the industrial relevance of Codeless3D is that the envisioned approach will offer SCALABILITY, since *"...it could cater both small indie teams and large game studios in the future when more features are developed..."* (P16). In addition, they expressed that Codeless3D *"...could offer a streamlined approach to game development, reducing the time and resources required to create 3D games..."* (P15), and *"...could make game development accessible to a wider audience, including artists, designers, and not only programmers..."* (P18), strengthening the industrial relevance. Regarding the readiness of the existing prototype tool, although the prototype does not include all the planned features of the envisioned approach, it demonstrated sufficient FEATURE COMPLETENESS to fulfill its primary objectives, a viewpoint supported by 12 participants (60%). Finally, regarding the pathway to industrial acceptance, 19 of them (90%) emphasized the significance of CONTINUOUS IMPROVEMENT. They argued that such a commitment would ensure that Codeless3D *"...could meet the needs and expectations of the game development industry, enhancing its acceptance..."* (P20) as well as that it *"...could be appealing to industrial stakeholders and hence it would be useful for them to adopt it in the future..."* (P8).

Nevertheless, some participants raised concerns regarding the absence of certain features in the prototype tool that are integral to the envisioned approach. Of particular significance was *"...the limited implementation of game mechanics and game elements..."* (P16), along with *"...the lack of linking to assets libraries, which limits the capability to add complex models, allowing only pre-defined models..."* (P11). Lastly, dissatisfaction was expressed with *"...the lack of drag and drop features indicating that this limitation diminishes the usability and potential of the proposed tool..."* (P12). Summarizing the vision of Codeless3D, participants viewed Codeless3D as industrially relevant due to its potential to streamline game development and broaden accessibility. In addition, while the prototype demonstrates sufficient feature completeness, continuous improvement was considered crucial for industrial acceptance.

VI. Limitations

Codeless3D poses several limitations that impact its usability and functionality. Firstly, the current version of the prototype tool implements limited game mechanics that restricts its ability to develop full-fledged games effectively. Additionally, the absence of asset libraries and drag-and-drop features further challenges its usability and potential, constraining users to use predefined models and options, diminishing overall user experience and creativity. Codeless3D lacks the capability to intuitively incorporate essential mechanics such as music, art, animation, etc., limiting its scope. Moreover, the manual input of coordinate and rotation information may lack intuitiveness, potentially complicating the design process. Finally, the complexity of adding more than one model results in scrolling down which further increases to the tool's complexity.

VII. Implications for Researchers and Practitioners

Implications for Practitioners. Considering the outcomes derived from this study, it is recommended that not only game programmers, but especially game / level designers are encouraged to embrace low-code tools such as Codeless3D and engage in experimentation to generate 3D platform games. Furthermore, the empirical evidence gathered in this study demonstrates that Codeless3D effectively reduces the design time and the overall development time for creating a 3D game.

Implications for Researchers. Based on the findings of the study, the motivation to develop an end-to-end approach for the generation of a 3D game, irrespective of the technical proficiency of potential users, with the primary objective of reducing design and development time, was a successful decision. However, as it is already mentioned the current prototype implements limited game mechanics, while the envisioned approach prioritizes the digitalization of GDD in a collaborative way for all members of the development team to create a 3D game. We note that with Codeless3D we do not aim at minimizing or eliminating communication among stakeholders but move it in a structured and online environment that will potentially improve collaboration rather than hinder it. Consequently, researchers are encouraged to delve deeper into studying and exploring the GDD, subsequently focusing on proposing innovative approaches for digitally transforming the GDD through the enhanced intuitive and low-code design tool.

Additionally, researchers are prompted to explore the potential benefits of integrating Codeless3D with PCG which would be an intriguing avenue to pursue. Such improvements would require a replication of the evaluation study to assess the usability of Codeless3D. Moreover, conducting larger-scale experiments comparing the workload and efficiency of using Codeless3D versus other traditional methods such as Unity3D, in control groups and experimental groups, would provide more insightful and valuable conclusions.

Finally, an interesting line of research that opens from supporting the formalization of game design process would be to extend the current state of practice with decision documentation. Inspired by the domain of software architecture [56], we believe that a methodology and a tool for documenting design decisions (e.g., "*why is health included?*" can be answered either by stating "*we wanted to imitate this particular game*", or by a more complex process, such as "*we included this as part of the procedural rhetoric [57] of the generated game*").

VIII. Threats to Validity

Construct Validity. The design and implementation of the usability evaluation instruments (i.e., questionnaire and focus group) may introduce biases or inaccuracies that affect the validity of the results. To mitigate this threat a well-established questionnaire for usability (SUS) was utilized. Additionally, during the focus group, explicit clarification was provided to address framing bias and emphasize the importance of both positive and negative evaluations in generating valuable research outcomes. Another potential threat to construct validity is mono-operation bias, which occurs when a single measurement is employed to assess the usability of the tool. To mitigate this threat, method triangulation was employed to gather data from multiple sources. By employing this approach, construct validity concerns were addressed by offering a comprehensive and multifaceted assessment of the construct. This reduced reliance on a single measure and ultimately enhanced the overall validity of the findings.

External Validity. The extent to which our validation findings can be generalized is influenced by the limited sample size utilized in the study, posing a potential threat to external validity. However, this concern is alleviated by the existing literature [54], which suggests that the SUS yields reliable results even with smaller sample sizes. Nonetheless, to enhance the external validity of future work, it would be advantageous to include a more diverse and representative sample from the target population of interest.

Reliability. The process of open coding is susceptible to biases introduced by multiple researchers, which may result in decreased inter-rater reliability and compromise the consistency of the results. To mitigate this potential threat, a systematic approach was employed during the coding process, and detailed documentation of the process was provided to enhance transparency. Additionally, extensive peer review was utilized throughout the coding process to validate and verify the various data analyses conducted for the study. These measures were implemented to mitigate potential biases and strengthen the reliability of the findings.

IX. Conclusions

This paper presents an end-to-end approach for game development, specifically focusing on the introduction of a design tool. Codeless3D with its low-code functionalities facilitates users to produce small -sized and -complexity games. In addition, Codeless3D empowers users without programming expertise to generate an entirely new 3D game. To evaluate the usability of Codeless3D, a study was conducted, wherein effectiveness, efficiency, user satisfaction and vision were assessed. The results of the usability evaluation were promising, highlighting the strengths of the tool, but at the same time, some weaknesses were pointed out that need to be improved. Users enthusiastically embraced the concept of Codeless3D, expressing excitement to use it and create games. Concerns included limited implementation of game mechanics and absence of drag-and-drop features, which affect usability and potential. Although Codeless3D is still in its initial stage, it can be asserted that it is suitable for 3D game generation, effectively reducing both design and development time, as the users involved in the study expressed overall satisfaction with the tool.

References

[1] G. Delmestri, F. Montanari, and A. Usai, "Reputation and strength of ties in predicting commercial success and artistic merit of independents in the Italian feature film industry," *Journal of Management Studies*, vol. 42, no. 5, pp. 975-1002, 2005. doi:10.1111/j.1467-6486.2005.00529.x.

[2] P. W. Atmaja and R. Parlika, "A Preliminary Study on Integrating Procedural Content Generation into Game Development Process," *IJCONSIST JOURNALS*, vol. 1, no. 1, pp. 27-34, 2019. doi:10.33005/ijconsist.v1i1.8.

[3] S. Aleem, L.F. Capretz, and F. Ahmed, "Game development software engineering process life cycle: a systematic review," *Journal of Software Engineering Research and Development*, vol. 4, no. 6, 2016. doi: 10.1186/s40411-016-0032-7.

[4] C. Politowski, L. Fontoura, F. Petrillo, and Y. Guéhéneuc, "Are the Old Days Gone? A Survey on Actual Software Engineering Processes in Video Game Industry," in *2016 IEEE/ACM 5th International Workshop on Games and Software Engineering (GAS)*, Austin, TX, USA, 2016, pp. 22-28. doi:10.1145/2896958.2896960.

[5] J. Dormans, "Engineering Emergence: Applied Theory for Game Design", Ph.D. Dissertation, Amsterdam University of Applied Sciences, Amsterdam, 2012.

[6] M. Baldwin, "Game design document outline," 2005, [Online]. Available: https://pdfcoffee.com/baldwin-game-design-document-template-pdf-free.html.

[7] D. Callele, E. Neufeld, and K. Schneider, "A report on select research opportunities in requirements engineering for videogame development," *2011 Fourth International Workshop on Multimedia and Enjoyable Requirements Engineering (MERE'11)*, Trento, Italy, 2011, pp. 26-33, doi:10.1109/MERE.2011.6043942.

[8] M. Daneva, "Striving for balance: A look at gameplay requirements of massively multiplayer online role-playing games," *Journal of Systems and Software*, vol. 134, pp. 54-75, 2017. doi:10.1016/j.jss.2017.08.009.

[9] M. E. Paschali, C. Volioti, A. Ampatzoglou, A. Gkagkas, I. Stamelos, and A. Chatzigeorgiou, "Implementing game requirements using design patterns," *Journal of Software: Evolution and Process*, vol. 33, no. 12, 2012. doi:10.1002/smr.2399.

[10] K. Neil, "Game Design Tools: Time to Evaluate," in *Proc. of DiGRA Nordic Conference: Local and Global Games in Culture and Society*, Tampere, 2012.

[11] K. Salen and E. Zimmerman, *Rules of Play: Game Design Fundamentals*. MIT Press, ACM Digital Library, 2003, pp. 80.

[12] E. Adams, *Fundamentals of game design*, 3rd ed., New Riders Publishing, USA, 2014.

[13] M. S. O. Almeida and F. S. C. da Silva, "A systematic review of game design methods and tools," *Entertainment Computing*, vol 8215, J. C. Anacleto, E. W. G. Clua, F. S. C. da Silva, S. Fels and H. S. Yang, Eds. Berlin, Heidelberg: Springer, 2013, pp. 17-29. doi:10.1007/978-3-642-41106-9_3.

[14] S. Aleem, L.F. Capretz, and F. Ahmed, "Critical success factors to improve the game development process from a developer's perspective," *Journal of Computer Science and Technology*, vol. 31, no. 5, pp. 925–950, 2016. doi: 10.1007/s11390-016-1673-z.

[15] D. Callele, E. Neufeld, and K. Schneider, "Requirements engineering and the creative process in the video game industry," in *Proc. of the 13th IEEE International Conference on Requirement Engineering*, 2005, pp. 240-250. doi:10.1109/RE.2005.58.

[16] D. Callele, E. Neufeld, and K. Schneider, "A proposal for cognitive gameplay requirements," in *Proc. of the 5th International Workshop on Requirements Engineering Visualization*, 2010, pp. 43-52. doi:10.1109/REV.2010.5625658.

[17] M. G. Salazar, H. A. Mitre, C. L. Olalde, and J. L. G. Sánchez, "Proposal of Game Design Document from software engineering requirements perspective," *2012 17th International Conference on Computer Games (CGAMES)*, Louisville, KY, USA, 2012, pp. 81-85, doi:10.1109/CGames.2012.6314556.

[18] H. A. Mitre-Hernandez, C. Lara-Alvarez, M. Gonzalez-Salazar, and D. Martin, "Decreasing Rework in Video Games Development from a Software Engineering Perspective," in *Proc. of the 4th International Conference on Software Process Improvement*, 2016, pp. 295-304. doi:10.1007/978-3-319-26285-7_25.

[19] N. Shaker, J. Togelius, and M. J. Nelson, "Procedural Content Generation in Games," *Computational Synthesis and Creative Systems*, Springer Cham, 2016.

[20] J. Togelius, E. Kastbjerg, D. Schedl, and G.N. Yannakakis, "What is procedural content generation?: Mario on the borderline," in *Proc. of the 2nd Workshop on Procedural Content Generation in Games*, Association for Computing Machinery, New York, NY, USA, vol. 3, pp. 1–6, 2011. doi:10.1145/2000919.2000922.

[21] M. Cook and S. Colton, "Ludus Ex Machina: Building A 3D Game Designer That Competes Alongside Humans," in *Proc. of the International Conference on Innovative Computing and Cloud Computing*, 2014.

[22] M. Cook and G. Smith, "Formalizing Non-Formalism: Breaking the Rules of Automated Game Design," in *Proc. of the International Conference on Foundations of Digital Games*, 2015.

[23] J. Dormans, "The effectiveness and efficiency of model driven game design," in *International Conference on Entertainment Computing*, 542-548, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[24] T. Schaul, "A Video Game Description Language for Model-based or Interactive Learning," in *IEEE Conference on Computational Intelligence in Games (CIG)*, 2013, pp. 1–8. doi:10.1109/CIG.2013.6633610.

[25] D. Perez-Liebana, S. Samothrakis, J. Togelius, S. M. Lucas, and T. Schaul, "General Video Game AI: Competition, Challenges and Opportunities," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016, pp. 4335–4337. doi:10.1609/aaai.v30i1.9869.

[26] M. Treanor, B. Blackford, M. Mateas, and I. Bogost, "Game-o-matic: Generating videogames that represent ideas," in *Proc. of the 3rd Workshop on Procedural Content Generation in Games*, 2012, pp. 1-8. doi:10.1145/2538528.2538537.

[27] A. Summerville, C. Martens, B. Samuel, J. Osborn, N. Wardrip-Fruin, and M. Mateas, "Gemini: bidirectional generation and analysis of games via ASP," in *Proc. of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 14, no. 1, pp. 123-129, 2018. doi:10.1609/aiide.v14i1.13013.

[28] M. Kreminski, M. Dickinson, J. C. Osborn, A. Summerville, M. Mateas, and N. Wardrip-Fruin. "Germinate: a mixed-initiative casual creator for rhetorical games," in *Proc. of the Sixteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE'20)*, AAAI Press, vol. 16, no. 1, pp. 102–108, 2020. doi:10.1609/aiide.v16i1.7417.

[29] K. Compton and M. Mateas, "Casual creators," in *Proc. of the 6th International Conference on Computational Creativity, ICCC*, 2015, pp. 228-235.

[30] M. J. Nelson, S. E. Gaudl, S. Colton, and S. Deterding, "Curious users of casual creators," in *Proc. of the 13th International Conference on the Foundations of Digital Games (FDG '18)*. ACM, New York, NY, USA, vol. 61, pp. 1–6, 2018. doi:10.1145/3235765.3235826.

[31] M. Guzdial and M. O. Riedl, "Automated game design via conceptual expansion," in *Proc. of the 4th AAAI Conference on AIIDE'18*, vol. 5, pp. 31–37, 2018. doi:10.1609/aiide.v14i1.13022.

[32] M. Guzdial and M. O. Riedl, "Conceptual Game Expansion," in *IEEE Transactions on Games*, vol. 14, no. 1, pp. 93-106, 2022. doi:10.1109/TG.2021.3060005

[33] M. Charity, A. Khalifa, and J. Togelius, "Baba is Y'all: Collaborative Mixed-Initiative Level Design," in *IEEE Conference on Games (CoG)*, Osaka, Japan, 2020, pp. 542-549, doi:10.1109/CoG47356.2020.9231807.

[34] N. R. Sturtevant, N. Decroocq, A. Tripodi, C. Yang, and M. Guzdial, "A demonstration of anhinga: a mixed-initiative EPCG tool for snakebird," in *Proc. of the Sixteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE'20)*. AAAI Press, vol. 49, pp. 328–330, 2020. doi:10.1609/aiide.v16i1.7451.

[35] N. Shaker, M.H. Shaker, and J. Togelius, "Ropossum: An Authoring Tool for Designing, Optimizing and Solving Cut the Rope Levels," in *Proc. of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 9, no. 1, pp. 215-216, 2013. doi:10.1609/aiide.v9i1.12611.

[36] S. Risi and J. Togelius, "Increasing generality in machine learning through procedural content generation," *Nature Machine Intelligence*, vol. 2, no. 8, pp. 428-436, 2019. doi:10.1038/s42256-020-0208-z.

[37] G. Smith, "Understanding procedural content generation: a design-centric analysis of the role of PCG in games," in *Proc. of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. Association for Computing Machinery, New York, NY, USA, 2014, pp. 917-926. doi:10.1145/2556288.2557341.

[38] J. Togelius, T. Justinussen, and A. Hartzen, "Compositional procedural content generation," in *Proc. of the 3rd workshop on PCG'12*. ACM, New York, NY, USA, pp. 1–4, 2012.

[39] Y. Luo, P. Liang, C. Wang, M. Shahin, and J. Zhan, "Characteristics and challenges of low-code development: the practitioners' perspective," in

Proc. of the 15th ACM/IEEE Int. Symposium on ESEM, 2021, pp. 1-11. doi:10.1145/3475716.3475782.
[40] A.C. Bock and U. Frank, "Low-code platform," *Business & Information Systems Engineering*, vol. 63, no. 6, pp. 733-740, 2021. doi:10.1007/s12599-021-00726-8.
[41] A. Sahay, A. Indamutsa, D. Di Ruscio, and A. Pierantonio, "Supporting the understanding and comparison of low-code development platforms," in *Proc. of Euromicro 2020 Conf. Software Engineering and Advanced Applications*, 2020, pp. 171-178. doi:10.1109/SEAA51224.2020.00036.
[42] B. Konopka, K. Hönemann, P. Brandt, and M. Wiesche, "WizARd: a no-code tool for business process guidance through the use of augmented reality," *Demonstration & Resources Track*, Best BPM Dissertation Award, and Doctoral Consortium at BPM, 2022.
[43] P. Zikas et al., "MAGES 4.0: Accelerating the World's Transition to VR Training and Democratizing the Authoring of the Medical Metaverse," in *IEEE Computer Graphics and Applications*, vol. 43, no. 2, pp. 43-56, 2023. doi: 10.1109/MCG.2023.3242686,
[44] P. Fleck, A.S. Calepso, S. Hubenschmid, M. Sedlmair, and D. Schmalstieg, "RagRug: A Toolkit for Situated Analytics," in *IEEE Trans. on Visualization and Computer Graphics*, vol. 29, no. 7, pp. 3281-3297, 2023. doi: 10.1109/TVCG.2022.3157058.
[45] A. Torres, B. Kapralos, C. Da Silva, E. Peisachovich, and A. Dubrowski, "Moirai: A No-Code Virtual Serious Game Authoring Platform," *Virtual Worlds*, vol. 1, no. 2, pp. 147-171, 2022. doi:10.3390/virtualworlds1020009.
[46] J. C. Osborn, N. Wardrip-Fruin, and M. Mateas, "Refining operational logics," in *Proc. of the 12th International Conference on the Foundations of Digital Games (FDG '17)*. ACM, New York, NY, USA, Article 27, pp. 1–10, 2017. doi:10.1145/3102071.3102107.
[47] J.C. Osborn, D. Lederle-Ensign, N. Wardrip-Fruin, and M. Mateas, "Combat in Games," *International Conference on Foundations of Digital Games*, 2015.
[48] J.C. Osborn, B. Lambrigger, and M. Mateas, "HyPED: Modeling and Analyzing Action Games as Hybrid Systems," in *Proc. of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 13, no. 1, pp. 87-93, 2021. doi:10.1609/aiide.v13i1.12937.
[49] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering: Guidelines and Examples*, John Wiley & Sons, 2012.
[50] V. Basili, G. Caldiera, and D. Rombach, "The Goal Question Metric Approach," in *Encyclopedia of Software Engineering*. Wiley & Sons, 1994, pp. 528-532.
[51] E. Frokjaer, M. Hertzum, and K. Hornbaek, "Measuring Usability: Are Effectiveness, Efficiency, and Satisfaction Really Correlated?," in *Proc. of the CHI*, vol 2, 2000.
[52] J. Brooke, "SUS - A quick and dirty usability scale," *Usability Evaluation In Industry*, 1st ed., vol. 189, CRC Press, 1996, pp. 4–7.
[53] A. Bangor, P.T. Kortum, and J.T. Miller, "An Empirical Evaluation of the System Usability Scale,", *International Journal of Human-Computer Interaction*, vol. 24, no. 6, pp. 574–594, 2008, doi:10.1080/10447310802205776.
[54] T.S. Tullis and J.N. Stetson, "A Comparison of Questionnaires for Assessing Website Usability," in *Proc. of the Usability Professionals Association (UPA)*, Minneapolis, MN, USA, pp. 7–11, 2004.
[55] S. Elo and H. Kyngäs, "The qualitative content analysis process," *Journal of Advanced Nursing*, vol. 62, no. 1, pp. 107-115, 2008. doi:10.1111/j.1365-2648.2007.04569.x.
[56] C. Manteuffel, D. Tofan, P. Avgeriou, H. Koziolek, and T. Goldschmidt, "Decision architect - A decision documentation tool for industry" *Journal of Systems and Software*, vol. 112, pp. 181-198, 2016. doi:10.1016/j.jss.2015.10.034.
[57] I. Bogost, "*Persuasive Games: The Expressive Power of Videogames*," The MIT Press, 2010.