Implementation and Analysis of Reusability Framework Design for Event User Interface Component in Phaser 3

Ahmad Arsyel Abdul Hakim[1], Dana Sulistiyo Kusumo[2], Jati H. Husen[3]
[1,2,3]School of Computing, Telkom University
Bandung, Indonesia
[1]ahmadarsyel.ah@gmail.com, [2]danakusumo@telkomuniversity.ac.id, [3]jatihusen@telkomuniversity.ac.id

***Abstract*— The growth of game development is now widely supported by software such as game engines and game frameworks. Phaser 3 is a popular HTML5 based game framework on the browser platform. However, Phaser 3 does not facilitate managing code of User Interface (UI) event component. Sometimes a game developer discovers that UI event component codes, such as buttons or joysticks that have been created, must be regenerated on subsequent projects with the same function. In this research, we proposed a reusability framework for Phaser 3 by combining the factory method and the singleton pattern in managing UI event components. The aim is to make developer productivity increased so that there is no need to repeat the algorithm or code that has been created previously. Our results show that the Phaser 3 UI standard has less a reusability value of 64,241%, while the proposed framework that was built has relatively higher reusability, that is equal to 84,576%. Game developers can add the combination of design patterns, as good software development practices, into an existing game framework to achieve code reusability without major changing on the existing game framework. This approach can also be used for the creators of game framework to use design patterns from the outset of development of game framework.**

***Index Terms*—reusability, Phaser 3, framework, component, user interface**

I. Introduction

Currently there is a lot of software, such as game engines and frameworks whose role is to support the development of games with various features. Game engine is a collection of systems tasked with rendering graphics, simulating physics, audio, and data structures of game objects, so that game developers can focus on game mechanics [1]. The terms game engine and game framework are often used together [2]. The distinct difference between game engine and framework is an IDE (Integrated Development Environment). Phaser 3 is an easy, stable, and popular HTML5 game framework on the browser platform. This popularity based on GitHub [1], that has the stars and used by higher than the most of other HTML5 game framework. Browser is a flexible platform, because it supports almost all of operating systems. The game that made for the browser can be played on many platforms. Gupta [3] explained that the points that must be considered for the success of a game can be seen from the ease of player control over the game [4]. A player has control over the game, so the interaction must be in accordance with the function. A player control means related to the User Interface (UI) of the game. Symbolic control [5] becomes the context in player control, which mimics the relationship between the actual movements performed by the player and related movements in the game, represented by the avatar. However, Phaser 3 does not facilitate managing UI event component code. A game developer will find that the UI event component code that was created in a previous project, is rebuilt with the same function in another project. The object-oriented concept can be utilized with functions that are packaged as libraries that will increase productivity. Reusability become important for game projects to manage UI event component codes. A reusable framework design [6] will be useful if the class structure is arranged with design patterns [7]. Our research problem is "**How to build an object-oriented framework for the UI event component in Phaser 3 by applying the principles of a reusable framework and analyzing the factors that influence it"**.

This paper is organized as follows. Section 2 provides literature review on the selected tools and methods, which used in our framework. In Section 3, an overview of design system for framework environment is explained. Section 4 highlights the evaluation of metric result and result analysis, where Section 5 provides conclusions and future research improvement.

II. Literature Review

The literature covers the tools, methods, event user interface component and metric calculations for analysis evaluation.

A. Phaser 3

Phaser 3 is a free desktop and mobile framework for HTML5 games on Canvas and WebGL in browsers. Phaser 3 was created by Richard Davey using the TypeScript. It is an open-source project, so that all developers can contribute to development. The appearance of hybrid app [8] development makes HTML5 games can be used as a distribution package and run on various native platforms such as Android and iOS. In Table I shows the Phaser 3 user data on GitHub.

TABLE I
Detailed statistics on Phaser 3 data on GitHub

Used by	Watch	Star	Fork	Contributor
9100	1300	26500	6100	433

[1]https://github.com/photonstorm/phaser

B. Framework

Frameworks are reusable libraries of modules that are used as work steps to solve certain problems or can be a collection of abstract classes for application design [9]. Frameworks can be constructed from other frameworks.

1) Scenario driven design: It is a method for building API frameworks, starting with the design of features that we use most frequently for frameworks [9]. Writing the framework design process begins with an example of using the code, then creates the object model. This method can see directly the developer side in using the framework, so it is easier to understand.

2) Criterion of good framework: There is a criterion [9] to be good quality of framework. There is a criterion to be a good quality of the framework that is (i) simple, (ii) well designed, (iii) a review of the design, (iv) learning existing framework, (v) designed to be expandable and (vi) consistent.

3) Design Pattern: The term design pattern arises as a result of problems that often occur among developers in software development [10]. The reusability value of a system will be higher when the design problem solved with the appropriate [11] design pattern. The design patterns grouped into three different objectives, that is creational, structural, and behavioral patterns [7].

(a) Factory Method
Factory Method provides an interface for creating objects, depending on the implementation of the subclass that to be created. Factory Method allows the class to defer instantiation of its subclasses [7]. Factory method is a group of creational patterns that can decouple the process of instancing objects by relying on the abstraction [14]. For example, the client does not need to know the creator class therefore the creator will delegate the creation of objects for the product.

(b) Singleton
Singleton ensures that a class has only one instance and that it is globally accessible to that object [7]. Singleton is a group of creational patterns that can ensure that there is only one instance during the program lifecycle. Singleton makes shared resources or variables easily accessible with the same results in any class [7].

C. Event User Interface Component

Event User Interface (UI) component is an interface object [15] that is given to the user on a device so that it can interact through input devices, such as a mouse, keyboard, touchscreen, and others. Components in the software can be set up for placement and function [16], such as buttons and joysticks.

D. Reusability Metric

Reusability means that the component properties of existing software allow it to be reused [13]. A software becomes reusable if the code, design, documentation, and components are also reusable [6]. Metric means the measurement of the object being observed. Reusability Metric is a measurement of software objects that affect a component so that it becomes reusable.

Based on literature studies that have been carried out [6][13], many factors influence the metric of reusability. However, if every entity in the software is calculated, it will only make the calculation worse [12] compared to focusing only on the main aspects of the characteristics in assessing reusability.

The main aspects of measuring the reusability metric for the framework must at least be eligible [13]: (i) Generality that calculated by GC (Generality of Class) metrics, (ii) Understandability that calculated with CBO (Coupling Between Object) and LCOM (Lack of Cohesion Method) metrics, (iii) Portability that calculated by CBO metrics, (iv) Maintainability that is accomplished by calculating CBO and NOC (Number of Children) metrics in class, and (v) Documentation that is measured with four other aspects, such as: (a) amount of documentation, (b) quality, (c) completeness, and (d) legal. Note the detailed information in Table II.

To assess the overall class component of the framework against the reusability (R_C), the value based on the Reusability Metric developed by Nyasente [13] is defined as Equation 1 below:

$$R_C = Generality + Understandability + Portability + Maintainability + Documentation \quad (1)$$

$$R_C = w1.Generality + w2.Understandability + w3.Portability + w4.Maintainability + w5.Documentation \quad (2)$$

w_i = weight in the aspect that affects with total value is one
$w_1 = 0.1$, $w2 = 0.2$, $w3 = 0.1$, $w4 = 0.2$, $w5 = 0.4$

$$Generality = GC$$
$$Understandability = \frac{CBO+LCOM}{2} \quad (4)$$
$$Portability = CBO$$
$$Maintainability = \frac{CBO+NOC}{2}$$
$$Documentation = \frac{Quantity + Legal + Completeness + Amount}{4}$$

The weights for w_1 to w_5 in Equation 2 are obtained from attributes of the factors that affect reusability. The R_C value will determine the reusability framework of the class and method.

TABLE II
OBJECT METRICS DETAIL THAT IS CALCULATED ON THE SOFTWARE COMPONENT [13]

Assessed factors	CBO	NOC	GC	LCOM
Definition	The number of classes that are connected to other classes	The number of subclasses of a class	The level of generalization component is determined by the level of abstraction	The value to which a function is related and bound to another function
Relation	The lower of CBO value, the more reusable	The lower of NOC value, the more reusable	The higher of GC value, the more reusable	The higher of Cohesiveness, the more reusable
Calculation (*n*)	The ratio of the number of classes connected to class *X* divided by the total class-bound	The ratio of the number of subclasses is just below the superclass hierarchy to the total inheritance form	The ratio of level divided by total levels (the deeper class, the lower its grade level)	Find the combination value of each method in the variables in the class that results in an empty (P), and non-empty (Q) set. The value of LCOM is 0 if $P < Q$ and become $P - Q$ for the other else
Range value	0 - 1	0 - 1	0 - 1	0 - 1

III. SYSTEM DESIGN

The following is the overview and workings of system design applied to the proposed framework.

A. The PhaserUIComponent Framework

The system was built into a framework called PhaserUIComponent by extending Phaser 3 and was developed on a personal computer OS Windows 10. This framework is a solution to facilitate and cover the shortcomings of Phaser 3 in optimizing the use of event UI component code that is reusable for player control.

The selection of the Phaser 3 framework is based on its ease and popularity in the field of HTML5-based game development. This popularity is evidenced by the GitHub data statistics in the Table I.

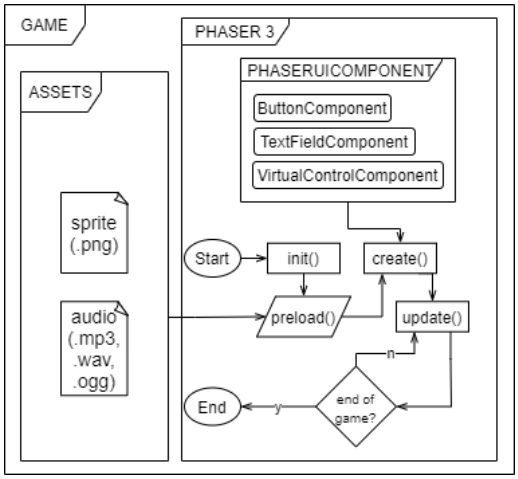

Fig. 1. Architectural Design

The architectural design in Figure 1 shows the relation of PhaserUIComponent to Phaser 3. PhaserUIComponent has three main classes based on the symbolic control approach [5], namely ButtonComponent, VirtualControlComponent, and TextFieldComponent for event UI component needs. Then, the results will be displayed on Canvas or WebGL, according to the wishes of the developer in the render configuration.

In general, the loop system in the Phaser 3 framework is the process of initializing objects from assets (Preload), variables (Init), rendering (Create), and detecting state (Update) which can be in the form of user input or other related conditions to update the position or status of the object.

The framework API is designed with a scenario-driven design method to help define the right API for the framework. This method approach is carried out so that the designer can see the results directly from the developer's perspective when the API is used [9]. The framework class structure is built by combining two design patterns that are the factory method and the singleton pattern.

The factory method can separate (decouple) the process of instancing objects from other objects that use them by relying on the form of abstraction [14]. This separation will make the design easier to develop and reuse in other applications in the form of a framework [7].

The use of singleton can encapsulate the system to be independent, simple, and consistent from calling the API in wrapping the framework so that reusability patterns can be created [17]. The system becomes independent because the framework can be accessed through the singleton without being associated with other classes. The singleton makes the framework simple and consistent because the developer will easily remember how to access to create UI object components in any class through the singleton class.

Design patterns are solutions to specific problems [7]. The combination of singleton and factory method patterns can encompass the framework, so that the API can be used globally and reusable, with the details of making objects that are separated by the abstraction class.

B. Game Development Process

The initial stages of game development (see Figure 2) have assets in the form of sprites and audio. UI that has been available or created is then implemented by the developer. The PhaserUIComponent Framework can be placed in the game folder structure (generally in *./project directory/src*). After that, the developer can adjust the initial configuration which would be used in the game. The framework can be used directly according to game requirements related to event UI components for player control.

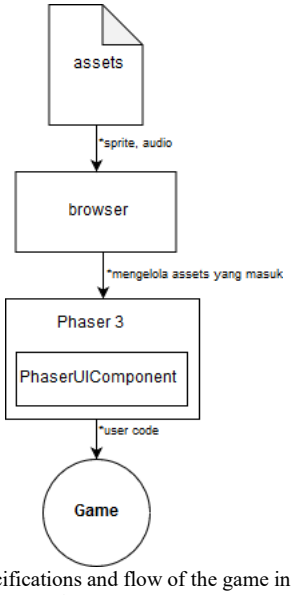

Fig. 2. The specifications and flow of the game in Phaser 3 with PhaserUIComponent

IV. Evaluation

This section consists of evaluation metrics and analysis for the framework and game programmer interviews to answer the research question.

A. Evaluation Scenario

1) Reusability Metric

Through the reusability metric calculation, the value will determine whether the framework is reusable or not. The choice of metric is due to the association between event UI components and software components [18], and their focus on the characteristics of coherent components and the effect on reusability [13].

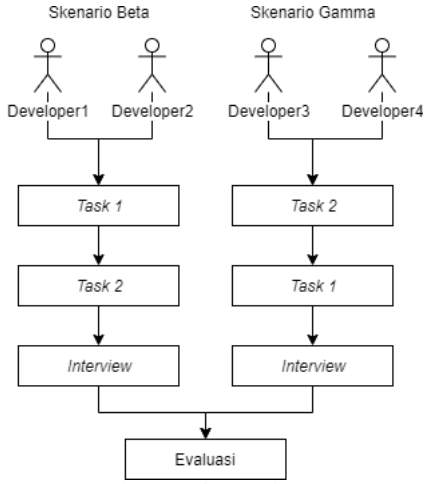

Fig. 3. Task scenario

2) Developer Interviews

This scenario is based on the use of PhaserUIComponent directly by the developer by doing selected tasks related to game development by coding UI event components. Scenarios are designed into two types of scenarios with different task sequences (see Figure 3) to avoid bias from developer tendencies that are affected due to the use of the previous framework.

The contents of the first task are to ask the developer to make a game with the code component of the Phaser 3 standard UI event, while the contents of the second task using the PhaserUIComponent framework.

B. Evaluation Analysis

1) Result and Analysis (Reusability Metric)

The first test scenario is performed on each class and the method of the PhaserUIComponent framework and the Phaser 3 UI standard. The test focuses on the level of reusability in framework UI event component. The results of the evaluation of each class component, method, and documentation, then look for the RC value on things that affect aspects based on Equation 1.

The following Table III is the detailed information needed based on the class diagram structure to assess reusable components, which is the total class that has an inheritance, the total of coupled class, and the depth level of the inheritance.

TABLE III
DATA FOR COMPONENT CALCULATIONS IN CLASS

Framework	Inheritance	Total coupled	Total level
PhaserUIComponent	7	11	2
Phaser 3 (Standard UI)	1	2	2

TABLE IV
AGGREGATION OF ASPECT VALUES THAT AFFECTED

Framework	PhaserUIComponent	Phaser 3 UI
Generality [GC]	0.682	0.833
Undestandablility [CBO, NOC]	0.815	0.195
Portability [CBO]	0.818	0.333
Maintainability [CBO, LCOM]	0.864	0.5
Documentation [4 Aspects]	0.9	0.967

The evaluation results in Table IV represent the value of each class component, such as the attributes, methods, and documentation for the aspects that influence.

TABLE V
THE COMPARISON OF THE REUSABILITY METRIC ON THE EVENT UI COMPONENT

	The Aspects	PhaserUIComponent	Phaser 3 UI
R_C	Generality	0.068	0.083
	Understandability	0.163	0.039
	Portability	0.082	0.033
	Maintainability	0.173	0.1
	Documentation	0.36	0.387

The RC results in Table V represent the reusability of the framework after being multiplied by its metric weight to the aggregation value.

The values for Generality and Documentation aspects of the Phaser 3 UI standard are superior, but not all aspects of reusability support. The total number reusability for PhaserUIComponent (0.84576) is better than the Phaser 3 UI standard (0.64241), where the maximum ratio value is one. Thus, it can be concluded that the reusability component in the PhaserUIComponent is relatively high, with a reusable percentage value of 84,576%.

2) Result and Analysis (Developer Interviews)

The results of the interviews conducted four respondents as a sample of developers who use the framework. The respondents have been briefly interviewed and are known to have a background in the game programmer and Phaser 3 users. The researcher interviewed three game programmers experienced from an Indonesian game studio and one student to participate in proposed framework usage for result and analysis. Then the qualitative results and their relevance are analyzed based on the reusability metric. There is eight-question that given to the developers.

TABLE VI
DEVELOPERS BACKGROUND

Developer	Background	Experience	Skill
1	Game Programmer	3 months	Medium
2	Informatics Student	4 months	Newbie
3	Game Programmer	3 months	Medium
4	Game Programmer	13 months	Medium-Advance

The first-three question is to discuss the ability and experience of Phaser 3 developers in making games. It can be seen from Table VI, where the respondent is a programmer with sufficient and experienced ability.

The fourth question discusses the ease of developers in implementing and managing UI event components using the PhaserUIComponent framework. Based on the developer responds, that implementing UI event components with the PhaserUIComponent framework is easy. It can also be seen from the duration of management in Table VII, where the average duration of tasks that developers do in using PhaserUIComponent can be 2.3 times faster than without PhaserUIComponent.

TABLE VII
THE TASK DURATION OF DEVELOPERS

Developer	Task Completion Time	
	Task1	Task2
1	2	0.5
2	3	1
3	1.5	1
4	1	0.75

Although it tends to be easy and fast, the high skill of developers makes managing UI using PhaserUIComponent still said to be standard.

The fifth question discusses the developer's convenience in creating UI components using the PhaserUIComponent framework. According to the developer's response, the process of making UI event components is standard, because it is very much determined by the skill and experience of the developer, where the more experienced, the smaller the difference in the time scale for doing the task. So, some developers respond to the neutral ease of the PhaserUIComponent framework in making UI.

The sixth question discusses the reusability of the PhaserUIComponent framework in creating UI components. All developers responded positively that PhaserUIComponent has a good level of reusability. This is relevant to the metric that was calculated in first evaluation.

The seventh question is discussed in the PhaserUIComponent documentation, which is easy to read and understand in helping to create UI components in the game. The documentation tends to be easy to read and understand, which is in line with the task time completion with PhaserUIComponent. See Table VII, using PhaserUIComponent (Task2) is always faster, than without PhaserUIComponent (Task1).

The last question is discussed the success of the PhaserUIComponent framework in creating event UI components. It is concluded from the average task that the PhaserUIComponent framework was successful, a Phaser 3 extension framework in creating and managing is an easy, documented, and reusable for event UI components. It is concluded from the average task that the PhaserUIComponent framework was successful. This extension framework is easy in creating and managing, documented, and reusable for event UI components.

Our results indicate that the existing game framework (the Phaser 3) still has problem with code reusability. Therefore, this research aims to improve this problem by implementing good software development practices [19] combining two design patterns (factory methods and singleton). The use of combination of the design patterns in this research is different than previous research [20][21]. In the previous research [20][21], design patterns were recommended in the development of game frameworks not specifically targeting for increasing code reusability producing from the existing game framework. Therefore, this approach can also be used for the creators of game framework to implement design patterns [20][21], so that it can have more and full control of game framework components [19] and can achieve targeted quality attributes.

V. CONCLUSION

The measurement of the reusability metric was successfully performed on the PhaserUIComponent framework. The framework was built under scenario-driven design methods for defining APIs, as well as structured class structures with design patterns, which are factory methods and singleton patterns. The metrics aspects of reusability that affect software components include maintainability, portability, understandability, generality, and documentation.

Thus, from the calculation of these aspects, the reusability value for the Phaser 3 UI standard is 0.64241 (the percentage of reusable values is 64,241%) and the PhaserUIC component is higher, which is worth 0.84576, with a maximum ratio of values of 1 (84,576% reusable percentage).

The quality results obtained from interviews with developers prove that the PhaserUComponent framework has been reusable as a framework for managing UI event components in Phaser 3. As a result, development time can be faster, easier to create and configure, due to easy-to-understand documentation.

Our research shows evidence which useful for the creators of game frameworks. They can get benefit by combining design patterns to achieve certain quality attributes when develop a game framework.

For further research, it can be re-evaluated components or class structures that reduce the assessment of reusability. This extends the selection of UI event components, not only to player controls. So, that it can be developed for more functionality in the field of 2D game development.

REFERENCES

[1] P. Mishra and U. Shrawankar, "Comparison between Famous Game Engines and Eminent Games," *Int. J. Interact. Multimed. Artif. Intell.*, vol. 4, no. 1, p. 69, 2016, doi: 10.9781/ijimai.2016.4113.

[2] B. Cowan and B. Kapralos, "A survey of frameworks and game engines for serious game development," *Proc. - IEEE 14th Int. Conf. Adv. Learn. Technol. ICALT 2014*, pp. 662–664, 2014, doi: 10.1109/ICALT.2014.194.

[3] S. Gupta, P. Kaushik, and S. Dawn, "Game Play Evaluation Metrics," *Int. J. Comput. Appl.*, vol. 117, no. 3, pp. 32–35, 2015, doi: 10.5120/20538-2902.

[4] D. Herumurti, A. Yuniarti, W. N. Khotimah, I. Kuswardayan, F. Revindasari, and S. Arifiani, "Analysing the user experience design based on game controller and interface," *2018 Int. Conf. Signals Syst. ICSigSys 2018 - Proc.*, pp. 136–141, 2018, doi: 10.1109/ICSIGSYS.2018.8372653.

[5] L. Herrewijn and K. Poels, "Recall and recognition of in-game advertising: The role of game control," *Front. Psychol.*, vol. 4, no. JAN, pp. 1–14, 2014, doi: 10.3389/fpsyg.2013.01023.

[6] J. Barnard, "A new reusability metric for object-oriented software," *Softw. Qual. J.*, vol. 7, no. 1, pp. 35–50, 1998, doi: 10.1023/b:sqjo.0000042058.34876.c8.

[7] G. Erich, H. Richard, J. Ralph, and V. John, "Design Patterns: Elements of Reusable Object-Oriented Software," *Addison-Wesley Prof.* 1994.

[8] M. Q. Huynh, P. Ghimire, and D. Truong, "Hybrid App Approach: Could It Mark the End of Native App Domination?," *Issues Informing Sci. Inf. Technol.*, vol. 14, pp. 049–065, 2017, doi: 10.28945/3723.

[9] K. Cwalina and B. Abrams, "Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .NET Libraries," *Addison-Wesley*. p. 480, 2005.

[10] F. Eric, F. Elisabeth, S. Kathy, and B. Bert, "Head First Design Patterns," *Oreilly & Associates Inc*. pp. 529–558, 2004.

[11] S. Yacoub and H. Ammar, "Toward Pattern-Oriented Frameworks," *JOOP - J. Object-Oriented Program.*, vol. 12, no. 8, pp. 25–46, 2000.

[12] L. Westfall and C. Road, "12 Steps to Useful Software Metrics," *Proc. Seventeenth Annu. Pacific Northwest Softw. Qual. Conf.*, vol. 57 Suppl 1, no. May 2006, pp. 40–43, 2005.

[13] S. Olive, N. Prof, W. Mwangi, and S. Kimani, "A Metrics-based Framework for Measuring the Reusability of Object-Oriented Software Components," *J. Inf. Eng. Appl.*, vol. 4, no. 4, pp. 71–85, 2014.

[14] B. Ellis, J. Stylos, and B. Myers, "The factory pattern in API design: A usability evaluation," *Proc. - Int. Conf. Softw. Eng.*, pp. 302–311, 2007, doi: 10.1109/ICSE.2007.85.

[15] D. M. Hilbert and D. F. Redmiles, "Extracting usability information from user interface events," *ACM Comput. Surv.*, vol. 32, no. 4, pp. 384–421, 2000, doi: 10.1145/371578.371593.

[16] W. Jiancheng, L. Xudong, and L. Lei, "A model of user interface design and its code generation," *2007 IEEE Int. Conf. Inf. Reuse Integr. IEEE IRI-2007*, pp. 128–133, 2007, doi: 10.1109/IRI.2007.4296609.

[17] V. K. Kerji, "Abstract Factory and Singleton Design Patterns to Create Decorator Pattern Objects in Web Application," *SSRN Electron. J.*, pp. 68–70, 2011, doi: 10.2139/ssrn.3439925.

[18] S. Yacoub, H. Ammar, and A. Mili, "Characterizing a Software Component," *Proc. Second Work. Component-Based Softw. Eng. Conjunction with Int'l Conf. Softw. Eng.* , 1999.

[19] C. Politowski, F. Petrillo, J. E. Montandon, M. T. Valente, and Y. G. Guéhéneuc, "Are game engines software frameworks? A three-perspective study," *J. Syst. Softw.*, vol. 171, p. 110846, 2021, doi: 10.1016/j.jss.2020.110846.

[20] R. T. Figueiredo, "GOF design patterns applied to the Development of Digital Games," *SBC – Proc. SBGames 2015*, pp. 1–8, 2015.

[21] N. H. Barakat, "A framework for integrating software design patterns with game design framework," *ACM Int. Conf. Proceeding Ser.*, pp. 47–50, 2019, doi: 10.1145/3328833.3328871.