International Journal
of Applied Information Technology

http://journals.telkomuniversity.ac.id/ijait/

Zetcil: Game Mechanic Framework for Unity Game Engine

Rickman Roedavan [a, *], Agus Pratondo [b], Rio Korio Utoro [c], Apriyanti Putri Sujana [d]

[a,b,c,d] *School of Applied Science, Telkom University, Indonesia*

rikman@telkomuniversity.ac.id, pratondo@gmail.com, korioutoro@telkomuniversity.ac.id, putrisujana@telkomuniversity.ac.id

ARTICLE INFO

Received 23 April 2020
Revised 13 May 2020
Accepted 22 June 2020
Available online 7 July 2020

Keywords:
Game Mechanic Framework, Game Design Pattern, Unity Game Engine

ABSTRACT

Games are interactive multimedia products that require programming logic and graphic design capabilities in the development process. Game development using Unity Game Engine has become a primary subject in the Multimedia Engineering Technology study program, School of Applied Sciences, Telkom University. In the initial observation, it was found that more than 33% of students have difficulties in making a game using the C# language for Unity. The students understand the Game Design Pattern logic but having difficulty when applying it into programming codes. This research proposes Zetcil, a Game Mechanic Framework that simplifies the game development process for the Unity Game Engine. This framework turned most of the text-based programming commands into visual properties. Based on research for two years, it concluded that the Zetcil could increase 36% of student's confidence to develop games prototype rapidly.

* Corresponding author at:
School of Applied Science, Telkom University
Jl. Telekomunikasi No. 1, Terusan Buah Batu, Bandung, 40257
Indonesia
E-mail address: rikman@telkomuniversity.ac.id

ORCID ID:
- First Author: 0000-0003-3169-2663
- Second Author: 0000-0002-6976-7459

https://doi.org/10.25124/ijait.v3i02.2779
Paper_reg_number IJAIT000030211 2020

1. Introduction

Games are interactive multimedia applications whose primary purpose is for entertainment needs. It combines programming logic, artificial intelligence (AI), graphic design, audio and video processing, and storytelling [1]. Moreover, game mechanisms can also be used to build products in other fields, such as health, education, military, business, and manufacturing [2][3][4].

Game development has become a subject in several well-known universities in Indonesia, especially Telkom University. Research on Game Development Life Cycle (GDLC) becomes a critical knowledge to understand the stages of its development [5][6].

There are many research studies related to the game development process and life cycle [7]. The majority of these development models agree that the game development process can be divided into three main phases, namely pre-production, production, and post-production.

The game production is a phase that combines elements of logic and art [8]. This phase is very difficult to be mastered by one person at a time. That is why the concentration of learning game development in universities is divided into two major parts [9].

The first part contains a discussion of programming logic, including the implementation of programming codes. This section is discussed in the Informatics Engineering or Information Systems study program. While the second part contains a discussion of design and visualization, this section is discussed in the Animation or Visual Communication Design study program.

Multimedia Engineering Technology is one of the study programs at the Faculty of Applied Sciences, Telkom University. This study program is a combination of Informatics Engineering and Visual Communication Design with a focus on building interactive multimedia applications, one of which is games.

Based on preliminary surveys to 120 respondents using 1-5 rank (1 means not confidence, and 5 means very confidence), it was found that more than 33% of Multimedia Engineering Technology students had difficulty learning C# programming language. Around 44% understood quite well, and only 3.4% understood the programming language very well. The complete survey results can be seen in Figure 1.

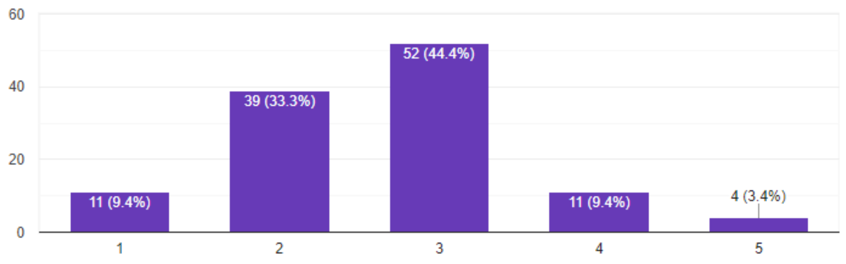


Figure 1 Preliminary Survey of implementing C# Programming Code using Visual Studio

Further interviews also found that if the majority of students have difficulty implementing mechanic game logic into programming code lines. This research tries to develop a Game Mechanic Framework for Unity Game Engine called Zetcil. This framework can simplify the game development process by turned most of the game programming logic into visual properties. Zetcil is designed using C# and can be used as a packaged plugin for Unity Game Engine Environment.

This paper comprises five chapters that are organized as follows. Chapter 2 talks about general explanations related to Interactive Multimedia development. Chapter 3 talks about the analysis and design of the Zetcil. Chapter 4 contains examples of the implementation and results of the Zetcil testing in lectures. Chapter 5 contains the matter of conclusions and discussion space for this research.

2. Game Design Pattern

Game design patterns are design patterns that are especially useful for making games [10]. Understanding game development patterns can accelerate the process of game development. Usually, all these patterns are implemented into lines of programming code.

This study tries to find game development solutions that can reduce the text implementation of programming code. One form of the pattern used in this study is the MVC (Model, Controller, View). This pattern, as shown in Figure 2, is not commonly used in most game development process [11] but are very useful to breaks down the basic components of a game based on a data model, input/interactions, and visualization.

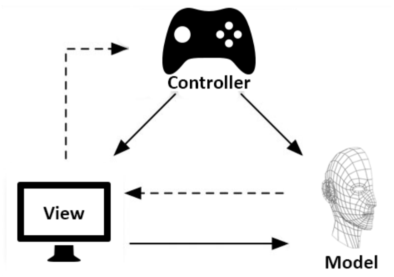


Figure 2 MVC as Pattern for Game Design

This pattern is somehow in line with some game design framework research that divided game elements into mechanics, dynamics, and aesthetics [12][13][14]. The game mechanic is algorithms, rules, objects, actions, and other game components, which are manipulated by game designers to create challenges for players. Player interactions with game mechanics will create dynamics. These dynamics, in turn, will affect the player's emotional experience or aesthetics [15].

This pattern also facilitates the process of developing a framework in the Unity environment. Unity divided the visual and game logic process separately by default. For the game logic process, Unity has two types of C# scripts, namely the Runtime script and the Editor script.

Runtime scripts are basic C# scripts that contain common logic and programming commands [16]. While Script Editor is a special script that functions to display variables, methods, or events in the form of visual properties, these scripts can be combined with basic Unity components to form a Prefab. Prefab is a visual representation of programming code, so this pattern fits perfectly with the profile of students' capabilities.

3. Proposed Solution

Zetcil is a proposed solution to address student's difficulty in implementing mechanic game logic into programming code lines. This framework is designed for

Unity Game Engine Environment and can display various types of the game logic in visual form. This should make it easier to create and design games without having to touch scripts or code programming directly [17].

The name Zetcil is taken from the word "jet," which means fast and "cil" which is a piece of the word "kecil" an Indonesian word of "small." So Zetcil can be interpreted as a collection of small elements that can help the game development process become faster.

3.1. Analysis and Design

This study tries to find a core pattern in every existing game mechanic and break it down into smaller elements [18][19]. This study breaks down the seven most popular game genres [20]. These game genres are First Person Shooter (FPS), Action Role Playing Games (ARPG), Side-Scrolling/Platformer, Top-Down Shooter (TDS), Real-Time Strategy (RTS), Mobile Battle Arena (MOBA), and Augmented Reality (AR) Battle Card.

The result is a game mechanic that can be broken down into two major parts, namely General Game Mechanic and Special Game Mechanic. General Mechanic Games are game functions that can be found in almost every type of game, such as health bar, score collection, movements, shooting, etc. While the Special Mechanic Game, which is a mechanic that is rarely found and is usually being a unique game characteristic, for example, the movement of a slingshot on Angry Bird or tapping on Flappy Bird.

These two types of game mechanics can break down into several small elements that have more specific functions. The game mechanic breakdown diagram is shown in Figure 3.

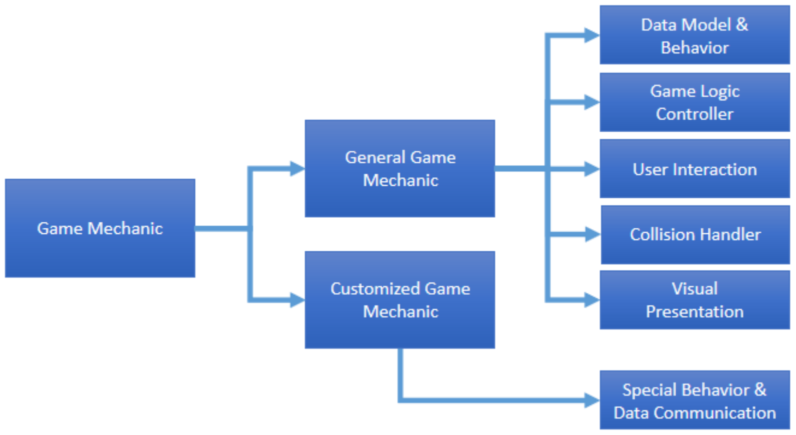


Figure 3 Game Mechanic Break Down

3.1.1. Data Model and Behavior

This element is a collection of basic data or variables that related to progress in the game, such as integer, float, string, or boolean. This data also has its own behavior according to its needs. For example, a timer element must be equipped with an increment/decrement behavior method, or integer must be equipped with add or subtract function.

3.1.2. *Game Logic Controller*

This element is a collection of basic programming logic. For example, check the value of a variable and perform other functions needed, looping an array of data, reading file text or xml configuration, playing sound/video files, or functions to manage scenes in the game.

3.1.3. *User Interaction*

This element is a collection of functions related to input interaction with the user. For example, functions to detect what keyboard keys are pressed, gestures detection for mobile input such as tap, swipe, rotate, and pinch, or voice recognition detection.

3.1.4. *Collision Handler*

This element is a collection of special functions to detect collisions between two or more sprites/meshes. This function is directly related to the main component in Unity and will check Rigidbody and Collider calculations for both 2D and 3D games.

3.1.5. *Visual Presentation*

This element is a collection of functions that manage the visual aspects of the game—for example, displaying various UI elements integrated with its own data, managing Camera movements according to the genre, or adding special effects in the game that triggered by user input or collision.

3.1.6. *Special Behavior and Data Communication*

This element is a collection of special functions that contain new game mechanic designs or modifications to third-party packages, such as Vuforia, Photon Engine, or Spatial OS. This element should have public variables or methods that can be integrated and communicate with other elements.

Briefly, how Zetcil works can be seen in Figure 4.

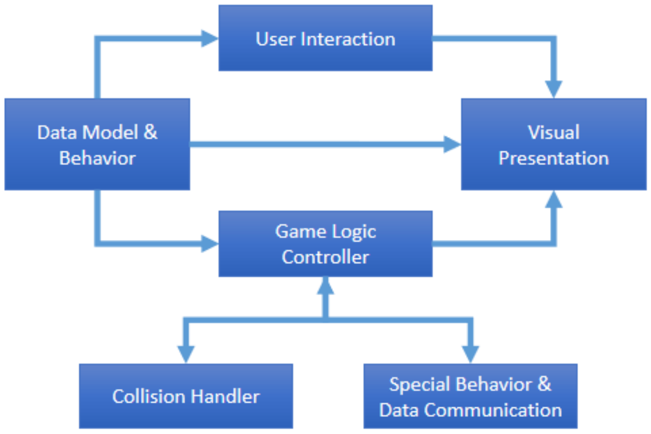


Figure 4 Zetcil Framework Data Flow

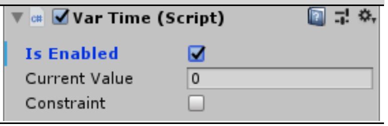
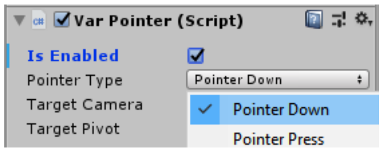
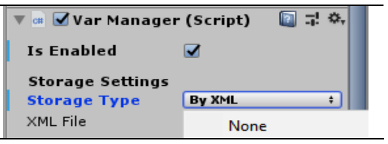
Zetcil uses the Data Model and Behavior element as the main input. This element can also be input into the User Interaction element and the Game Logic Controller element or directly display the value in the Visual Presentation group.

Meanwhile, Game Logic Controller will integrate another data form Collision Handler or Special Behavior and Data Communication element and show it to visual Presentation, respectively.

3.2. Implementation

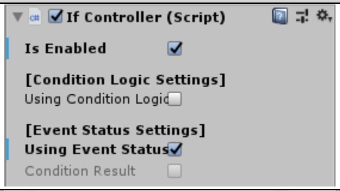
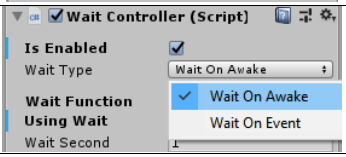
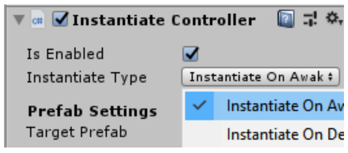
Zetcil Game Mechanic Framework developed using C#. At the current stage, it is divided into six groups consisting of 167 Prefabs, 150 Runtime scripts, and 65 Editor scripts. The following Table 1 shows some examples of Zetcil Prefabs in the Data Model & Behavior group that functions to manage data.

Table 1 Zetcil Framework: Data Model and Behavior

No	Data Model and Behavior	Screenshot
1	VarTime Prefab functions to store and calculate float variables to show the time/timer function.	 Var Time (Script) Is Enabled ✓ Current Value 0 Constraint
2	VarPointer Prefab functions to detect input from the mouse, read coordinate values on the screen and 3D world, and determine what objects are detected at that point using Raycast	 Var Pointer (Script) Is Enabled ✓ Pointer Type Pointer Down Target Camera Target Pivot ✓ Pointer Down Pointer Press
3	VarManager Prefab functions to perform basic data management such as integer, string, float, or boolean in the form of binary or XML files.	 Var Manager (Script) Is Enabled ✓ Storage Settings Storage Type By XML XML File None

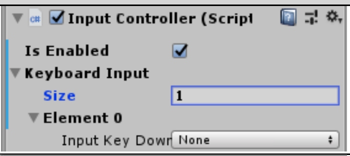
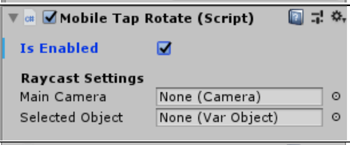
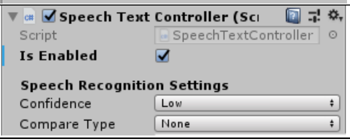
The following Table 2 shows some examples of Zetcil in the Game Logic Controller group that functions to manage basic logic in game programming.

Table 2 Zetcil Framework: Game Logic Controller

No	Game Logic Controller	Screenshot
1	If Controller Prefab functions to perform a visual version of if-then-else programming logic, this prefab can run events according to user needs.	 If Controller (Script) Is Enabled ✓ [Condition Logic Settings] Using Condition Logic [Event Status Settings] Using Event Status ✓ Condition Result
2	Wait Controller prefab functions to delay at a certain time before running other events that the user needs.	 Wait Controller (Script) Is Enabled ✓ Wait Type Wait On Awake Wait Function Using Wait Wait Second 1 ✓ Wait On Awake Wait On Event
3	Prefab Instantiate Controller functions to instantiate a GameObject at runtime. For example, for making bullet objects on the firing mechanics.	 Instantiate Controller Is Enabled ✓ Instantiate Type Instantiate On Awak Prefab Settings Target Prefab ✓ Instantiate On Av Instantiate On De

The following Table 3 shows some examples of Zetcil in the User Interaction group that function to manage input interactions with users

Table 3 Zetcil Framework: User Interaction

No	User Interaction	Screenshot
1	Prefab Input Controller functions to detect keyboards in three KeyDown, KeyPress, and KeyUp states and give choices on what events to run.	 Input Controller (Script) Is Enabled ✓ Keyboard Input Size 1 Element 0 Input Key Down None
2	Prefab Mobile Tap Rotate functions to detect a touch on mobile devices and to do rotations of certain objects on horizontal/vertical or both.	 Mobile Tap Rotate (Script) Is Enabled ✓ Raycast Settings Main Camera None (Camera) Selected Object None (Var Object)
3	Prefab Speech Controller functions to detect input in the form of sound and run certain events under voice commands that are successfully recognized.	 Speech Text Controller (Script) Script SpeechTextController Is Enabled ✓ Speech Recognition Settings Confidence Low Compare Type None

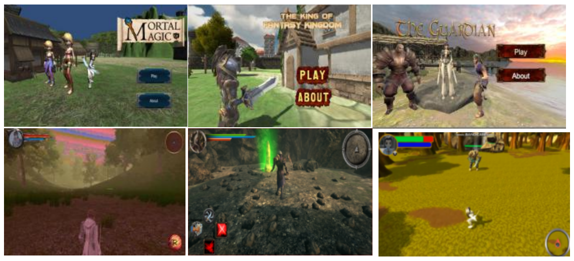
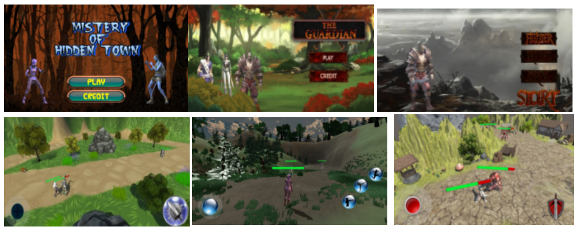
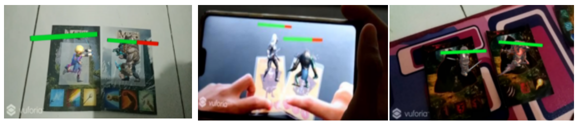
4. Experimental Result

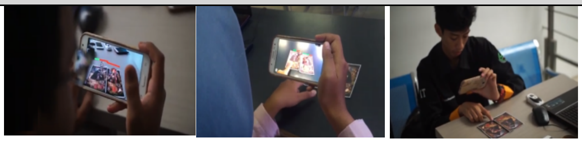
This study was conducted for two years to test the validity and quality of Zetcil. The subject is students of Multimedia Engineering Technology (2018-2019), Genesis Game Workshop (2018-2019), and international classes in collaboration with Universiti of Kuala Lumpur (2019).

The results show that although most target students do not have a strong programmer background, the target students can still build games according to the game criteria. Zetcil also can speed up the game development process for both groups and individual users.

The following Table 4 shows some games created by students.

Table 4 Games Built using Unity and Zetcil Framework

No	Game Screenshot	Description
1	 MORTAL MAGIC THE KING OF FANTASY KINGDOM PLAY ABOUT THE GUARDIAN Play About	This type of game is the final project of a basic game programming course that is done in groups with 3-4 students per group. The game itself is a PC-based Action Role Playing Game (RPG) game genre. All 3D models used free licensed models. Then, Animations created using Mixamo Animation Services. The main inputs for this game are the keyboard and mouse.
2	 MISTERY OF HIDDEN TOWN PLAY CREDIT THE GUARDIAN PLAY CREDIT START	This type of game is the second assignment for the advanced game programming course, which is done individually. The focus is only on developing mechanical Battle Battle Arena (MOBA) -like on the Android platform. The main inputs for this game are the Android direction pad.
3		This type of game is the final project of an advanced game programming course that is done in groups with 3-4 students per group. This game is an Android-based Augmented Reality (AR) Battle Card game. All 3D models used are free licensed models. Animations created using Mixamo Animation Services. This

No	Game Screenshot	Description
		game was built using the Vuforia SDK with the main input in the form of markers and virtual buttons.

This study also provides a questionnaire related to the level of confidence in developing basic games using the Unity Game Engine in 1-5 rank (1 means the respondent feels unconfidence, and five means the respondent feels confidence). This questionnaire was distributed to 120 respondents. The questions are:

Q1: How high is your confidence level in developing object-based game levels (plane, cube, stair, maze) using Unity Game Engine!

Q2: How high is your confidence level in developing terrain game levels (mountains, trees, grass, lakes) using the Unity Game Engine!

Q3: How high is your confidence level in developing character animations using Mixamo Animation Services and Animator Controller on the Unity Game Engine!

Q4: How high is your confidence level in developing games mechanic based on the Unity Game Engine using your own script!

Q5: How high is your confidence level in developing games mechanic based on the Unity Game Engine using the Zetcil Framework!

Figure 5-9 shows the result of Question Q1-Q5 respectively

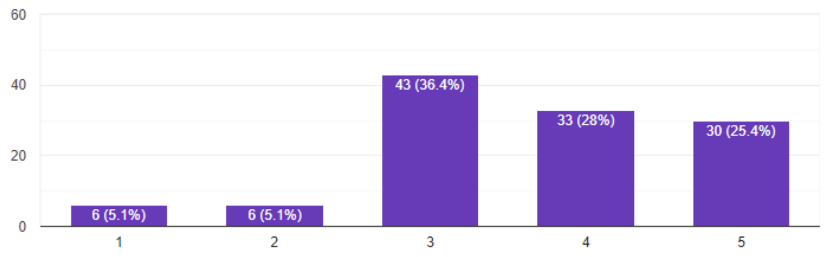


Figure 5 The Result of Question Q1

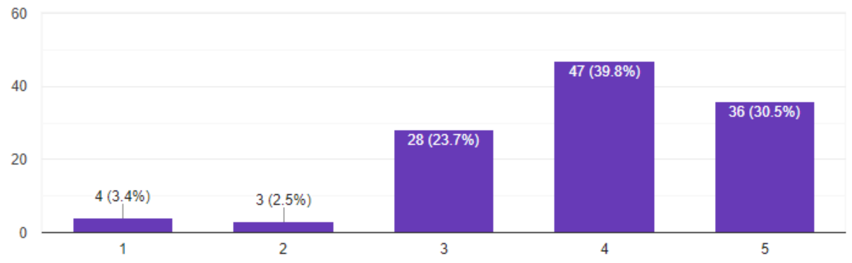


Figure 6 The Result of QuestionQ2

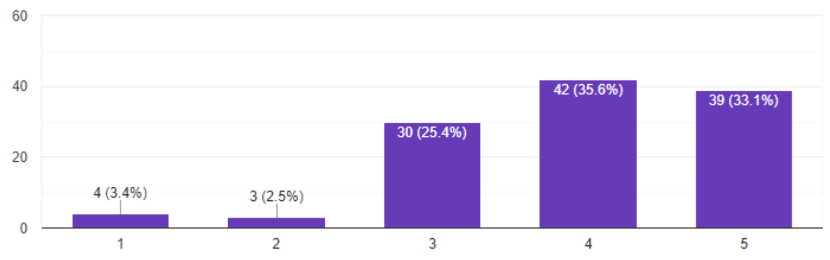


Figure 7 The Result of Question Q3

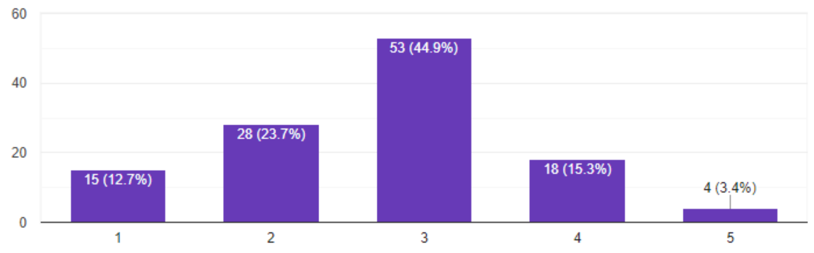


Figure 8 The Result of Question Q4

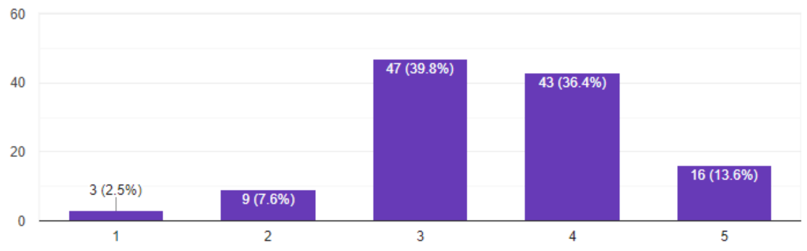


Figure 9 The Result of Question Q5

Questions Q1, Q2, and Q3 relate to the basic use of Unity to create visual aspects in games such as terrain, characters, and animation. Most of the students were confident and were able to do the game development process on this because it was done visually. Only less than 6% of students find it difficult to do this process.

While the questions Q4 and Q5 relate to making mechanics and logic in the game. These two questions produce answers that are inversely proportional. More than 30% of respondents were not capable in Q4 when requiring to make game logic using text-based (specifically the C # programming language). While in Q5, the majority of respondents were able to make game logic based on visuals using the Zetcil Framework. And only less than 10% of students still find it difficult to do this process.

The results produced an answer pattern that supports the hypothesis that the game mechanic framework based on visual properties can increase student confidence in building game prototypes when compared to text-based programming.

5. Conclusions

Zetcil Game Mechanic Framework is the proposed solution to simplify the game development process using the Unity Game Engine. Zetcil also can be used for rapid prototyping games for non-programmer users. At present, Zetcil has over 200 C# scripts and nearly 150 prefabs explicitly intended for game development.

The results of the experiments show there is a significant increase in Multimedia Engineering Technology student's confidence level in creating a game. However, further research is still needed to verify the ability of this framework to adopt new types of game mechanics, linkages with third-party script compatibility, and code stability in developing more complex games.

Bibliography

[1] Aleem, Saiqa & Capretz, Luiz & Ahmed, Faheem. "Game Development Software Engineering Process Life Cycle: A Systematic Review." *Journal of Software Engineering Research and Development.* 2016.

[2] Laamarti, Fedwa & Eid, Mohamad & El Saddik, Abdulmotaleb. "An Overview of Serious Games." *International Journal of Computer Games Technology*. 10.1155/2014/358152. 2014.

[3] Sun, Hanqiu & Ricciardi, Francesco & De Paolis, Lucio Tommaso. "A Comprehensive Review of Serious Games in Health Professions." *International Journal of Computer Games Technology*. 2014

[4] Darwesh, Aso, "Concepts Of Serious Game In Education." *International Journal Of Engineering And Computer Science*. 10.18535/Ijecs/v4i12.25, 2016

[5] A. Hendrick, "Project Management for Game Development" [Online]. Available: http://mmotidbits.com/2009/06/15/project-management-for-game-development/. 2009. [Accessed 18 March, 2020].

[6] H. M. Chandler, "Game Production Handbook" (Book style), Sudbury: Jones and Bartletts Publishers, 2010.

[7] Ramadan, Rido & Widyani, Yani. "Game development Life Cycle Guidelines." 95-100. 10.1109/ICACSIS.2013.6761558. 2013

[8] Blitz Games Studios, "Project Lifecycle," [Online]. Available: http://www.blitzgames studios.com/blitz_academy/game_dev/project_lifecycle. 2011. [Accessed 18 March, 2020].

[9] Boudreaux, Hollie & Etheridge, Jim & Kumar, Ashok. "Evolving Interdisciplinary Collaborative Groups in a Game Development Course." [Online]. Available: https://www.rit.edu/gccis/gameeducationjournal/evolving-interdisciplinary-collaborative-groups-game-development-course, 2011. [Accessed 18 March, 2020].

[10] Nystrom, Bob. "Game Programming Pattern." [Online]. Available from https://game programmingpatterns.com/design-patterns-revisited.html. 2014

[11] Ozkan, Eray, and Grove, Ralph. "The MVC Web Design Pattern." 7th International Conference on Web Information Systems and Technologies. 2011

[12] Hunicke, Robin & Leblanc, Marc & Zubek, Robert. "MDA: A Formal Approach to Game Design and Game Research." AAAI Workshop - Technical Report. 1. 2004.

[13] Schell. J. "The Art of Game Design". Morgan Kaufmann. 2008.

[14] Ralph, P. & Monu, K. "A Working Theory of Game Design. Mechanics, Technology, Dynamics, Aesthetics & Narratives". [Online]. Available from: http://www.firstperson scholar.com/a-working-theory-of-game-design. 2014/ [Accessed 20 March 2020].

[15] Walk, Wolfgang & Görlich, Daniel & Barrett, Mark. "Design, Dynamics, Experience (DDE): An Advancement of the MDA Framework for Game Design." 10.1007/978-3-319-53088-8_3. 2017

[16] Roedavan, Rickman. "Unity Tutorial Game Engine Revisi Kedua". Bandung. Penerbit Informatika. 2018.

[17] Roedavan, Rickman. "Construct2 Tutorial Game Engine". Bandung. Penerbit Informatika. 2017.

[18] Swanson, Richard A. "Theory Building in Applied Disciplines." San Francisco, CA: Berrett-Koehler Publishers, 2013.

[19] Frost, Brad. "Atomic Design Methodology". [Online]. Available from: https://atomicdesign. bradfrost.com/chapter-2/ 2016 [Accessed 20 March 2020].

[20] Statista, "Genre breakdown in the United States" [Online]. Available from: https://www.statista.com/statistics/189592/breakdown-of-us-video-game-sales-2009-by-genre/ 2018 [Accessed 20 March 2020].