Journal of Engineering Science and Technology
Vol. 16, No. 5 (2021) 4205 - 4213

DEVELOPMENT OF CLASS LIBRARY IN DOMAIN OF SCROLLING SHOOTER GAME

ADAM MUKHARIL BACHTIAR*,
DIAN DHARMAYANTI, RAKHMAT SABARUDIN

Department of Informatics Engineering, Universitas Komputer Indonesia, Indonesia
*Corresponding Author: adam@email.unikom.ac.id

Abstract

The purpose of this research is to develop a Class Library in the domain of Scrolling Shooter games. The built of this Class Library will be used to develop Scrolling Shooter games as the result of this research. In software engineering, there is a reuse concept that is used to reuse functional systems. One application of this concept is the class library. With the class library, programmers can directly call the functions they need without building them from scratch. This study also used the class library concept for the domain of the shooter scrolling game case. The development of a Class Library is done through seven phases, Domain Analysis, Frozenspot Analysis, Hotspot Analysis, Modelling Class Library, Implementation Class Library, Unit Testing, and Integration Testing, and those will be the main discussion of this research. Furthermore, this research's conclusion is expected to help game programmers develop Scrolling Shooter games without making the whole system functional from scratch. This research also contributes to software engineering concepts, namely: reusable code, abstraction, and class design.

Keywords: Class library, Domain analysis, Frozenspot analysis, Game scrolling shooter, Hotspot analysis.

1. Introduction

Based on data obtained from Newzoo (an intelligence market provider company in the global market), in 2017, the Indonesian gaming industry ranked 16th in the list of gaming markets in the world with several players reaching 43.7 million with potential income reaching around IDR. 11.9 trillion [1]. One of the popular game genres in Indonesia in January 2018 is Scrolling Shooter; it is noted that Scrolling Shooter won the top 10 on the Google Play Store in the January 2018 period, sorted by the Arcade game category. With this positive trend, game developers will automatically be interested in building a Scrolling Shooter game going forward.

The interview results with one of the programmers found that the problem in building a scrolling shooter game was that the programmer had to rebuild the functional system from scratch [2]. It caused the long development of the Scrolling shooter game, resulting in the game not being released according to the target time [3]. If you pay attention, the game industry always follows trends that require developers to release games at the right time.

Based on these problems, it is necessary to build a Class Library to help programmers build Scrolling Shooter games. A Class Library is a group of classes that provide the basic functionalities and can be used directly by programmers. With the Class Library, programmers can reuse functional requirements without having to build them from scratch. Therefore, a class library was created in this study to provide basic functionalities in accordance with the needs of programmers in building scrolling shooter games. This class library is an application of the reuse concept in software development.

2. Method

This research belongs to the type of applied research using a quantitative approach. Applied research aims to find a solution to deal with society or business organizations [4]. Meanwhile, the quantitative approach is used because research starts from a hypothesis and is objective. The methodology used in this study refers to the journal "Development of Class Library in Domain of Scrolling Shooter Game" by adding the implementation stages as seen in Fig. 1.

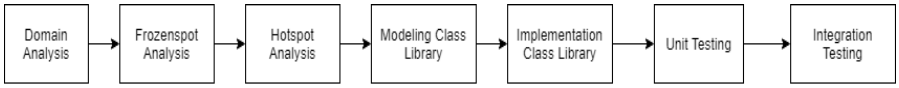

Fig. 1. Research methodology in this research.

This research methodology is that domain analysis was conducted to determine the characteristics of the scrolling shooter game application domain. The characteristics obtained are used to determine the scrolling shooter game domain's uniqueness from other game domains. Frozenspot analysis was carried out to determine the function slices of several samples of scrolling shooter game applications. Meanwhile, hotspot analysis is conducted to identify and define the points of flexibility (hotspots) of the scrolling shooter game application domain. Class library design is done to model hotspots into a modelling language, namely UML (Unified Modelling Language). The results of this modelling can simplify the implementation process into a programming language. Class library implementation is a step taken to change the class library design model into a

programming language. The unit testing phase is carried out to determine that each unit or function component in the class library can function as expected. Integration testing is done by directly testing each unit in the class library to create a scrolling shooter game prototype.

3. Results and Discussion

3.1. Domain analysis

Domain analysis can be done by determining the source of information used to identify the characteristics of the application domain. Several sources of information can be used, including domain experts, textbooks, domain standards, and previous application samples [5]. There are four sample Scrolling Shooter game applications from the Google Play Store that will assist the identification process. A characteristic of the Scrolling Shooter game used for this research is that players generally play as pilot fighter planes, which are assigned to withstand any enemy attacks [6]. The sample application consists of two types of vertical Scrolling Shooter games and two types of horizontal scrolling shooter games selected based on the highest number of downloads and the highest rating on the Google Play Store in the January 2018 period. The four sample game applications will be given symbols to facilitate calling the application name in the identification process, as shown in Table 1.

Table 1. Four sample game scrolling shooters.

Function	*Game*	Symbol
Space Shooter: Galaxy Attack	Vertical	G1
Galaxy Attack: Alien Shooter	Vertical	G2
Scrambler – Classic 80s Arcade Game	Horizontal	G3
R-TYPE	Horizontal	G4

Based on the identification results, four aspects become a review of the domain's characteristics: the camera viewpoint, game presentation, character control, and the game's basic mechanism. Here are the results of identifying the characteristics obtained from the four sample Scrolling Shooter game applications:

a. Camera Viewpoint

In G1 and G2 games, the camera's perspective is taken from the top to down (top-down view), where only the top side of the object is seen by the player when playing the game. While in the G3 and G4 games, the camera's perspective is taken from the side, where the player sees only the object's side. Viewpoints of the four games.

b. Game Presentation

The presentation or display style used in the G1 and G2 games is vertical scrolling. Vertical Scrolling is a display style where the game's background is seen moving automatically with a vertical orientation [7]. Meanwhile, in G3 and G4 games, the presentation used is horizontal scrolling. Horizontal Scrolling is a display style where the background in the game moves automatically with a horizontal orientation.

c. Character Control

The four sample Scrolling Shooter game applications that are played show that the player's character can move with an eight-way mechanism (right,

left, up, down, top right, bottom right, top left, and bottom left) on the *x* and *y* axes. The character movement controlled by the player is only limited to the screen area. There are two different mechanisms for controlling character objects, namely control with a fixed mechanism and control with a relative mechanism. In fixed mechanism control, the player's screen position has no distance from the character object being controlled. Whereas in the relative mechanism, there is a distance between the character object being controlled and the screen's position being touched.

d. Basic Mechanic

In the four sample games that are played, players must survive the pattern of attacks fired by the enemy to get the maximum score. To survive the enemy attack, players can destroy enemies by firing projectiles or avoiding incoming attacks. Also, there are power-ups that players can use to increase attack power. The four sample Scrolling Shooter game applications' characteristics will be used to identify the functions contained in the Scrolling Shooter game domain in the Frozenspot Analysis stage.

3.2. Frozenspot Analysis

Frozenspot analysis is divided into two stages, namely Function Identification and Frozenspot Identification.

a. Functional Identification

This stage is done to identify the collection or set of functions in the Scrolling Shooter game domain, based on four previous application samples. The resulting set of functions will be used to determine the Scrolling Shooter game domain's function slices. Namely, there are nine functions: Character Control, Scrolling Background, Spawn Object, Destroy Object, Move Object, Shoot Object With Pattern, Change Attribute, Display Attribute, and Save Data. The function identification table results show that there are nine sets of functions in the Scrolling Shooter game application domain and are used as a reference in determining the function slices.

b. Frozenspot Identification

This stage is carried out to determine the four previous Scrolling Shooter game application samples' sliced function. The set of functions that have been obtained will be used as a reference in determining the function slices. Frozenspot Identification results from four sample Scrolling Shooter game applications based on nine functions that have been obtained can be seen in Table 2. From the Frozenspot Analysis results, it can be seen that there are nine functions contained in the Scrolling Shooter game application domain. The nine functions are slices of the four sample Scrolling Shooter game applications G1, G2, G3, and G4. The sliced function results will be used to determine the hotspot that is done in the next stage.

Table 2. Frozenspot identification result.

Function	Game
Character Control	G1, G2, G3, G4
Scrolling Background	G1, G2, G3, G4
Spawn Object	G1, G2, G3, G4
Destroy Object	G1, G2, G3, G4

Function	Game
Move Object	G1, G2, G3, G4
Shoot Object with Pattern	G1, G2, G3, G4
Change Attribute	G1, G2, G3, G4
Display Attribute	G1, G2, G3, G4
Save Data	G1, G2, G3, G4

3.3. Hotspot Analysis

Hotspot analysis is divided into two stages, namely Hotspot Identification and Hotspot Definition [8].

a. Hotspot Identification

This stage is carried out to identify functions that can be used as hotspots in the Scrolling Shooter game application domain. The sliced function obtained from the frozen-spot analysis results can determine whether the function is suitable as a hotspot or not. In this study, Hotspot Identification uses the Blackbox method, where the function used as a hotspot is mandatory. Therefore, the hotspot identification results show that there are nine mandatory functions used as hotspots.

b. Defining Hotspot

This stage is carried out to define each function obtained from the Hotspot Identification results. Defining Hotspot is done by describing the behaviour performed by the function, using a hotspot card [9]. With the object model transformation rules, the hotspot card will be converted into object models in the form of class diagrams. The hotspot card from the Player Character Control function and the object model formed can be seen in Fig. 2.

***Hotspot* Name**
Character Control
Degree of Flexibility
☒ *adaptation without restart*
☐ *adaptation with by end user*

General Description
This function is used to move the character object in eight directions, based on the input in the form of a touch from the player.

Examples of implementation, at least two different situations
1. Control of character objects using a fixed mechanism, where the character object will always move to follow the position of the screen touched by the player without any difference in distance between the position of the character object with the position of the screen touched.
2. Character object control uses a relative mechanism, where the character object will always move to follow the position of the screen touched by the player with the possibility of the difference in distance between the position of the character object and the position of the screen touched.

Fig. 2. Hotspot card character control.

Based on the object model transformation rules, the hotspot design will be created by adding the hook method in a separate hook class. The transformation of the object model of the hotspot Controlling Player Character can be seen in Fig. 3.

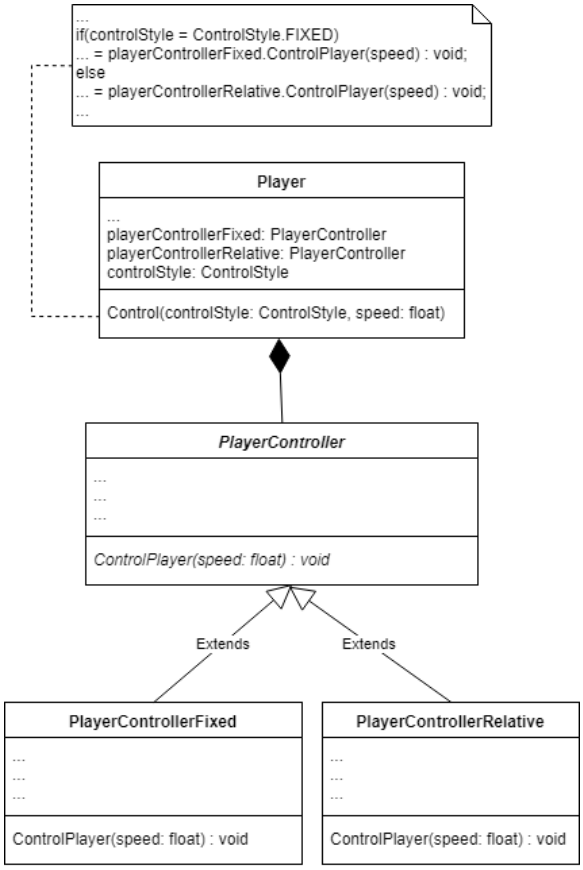

Fig. 3. Object transformation model hotspot character control.

3.4. Modelling class library

Modelling class library design consists of two stages, namely class design and package design.

a. Class Modelling

This class diagram will illustrate the relationships between classes of hotspot Controlling Player Character, where the class contains a method used to move character objects with eight directions of movement according to player control [10]. The class diagram of the hotspot controlling the player's character can be seen in Fig. 4.

b. Package Modelling

This stage is done to describe the package diagram of the Class Library system that will be built. The package diagram created functions to find groups of related classes wrapped in one container or package. The package design results show that nine packages will be built in the Scrolling Shooter class library, as follows.

- ScrollingShooter2D.Player
- ScrollingShooter2D.Spawner

- ScrollingShooter2D.Shoot
- ScrollingShooter2D.DisplayUI
- ScrollingShooter2D.Destroyer
- ScrollingShooter2D.Data
- ScrollingShooter2D.Backgroun
- ScrollingShooter2D.Attribute
- ScrollingShooter2D.Movement

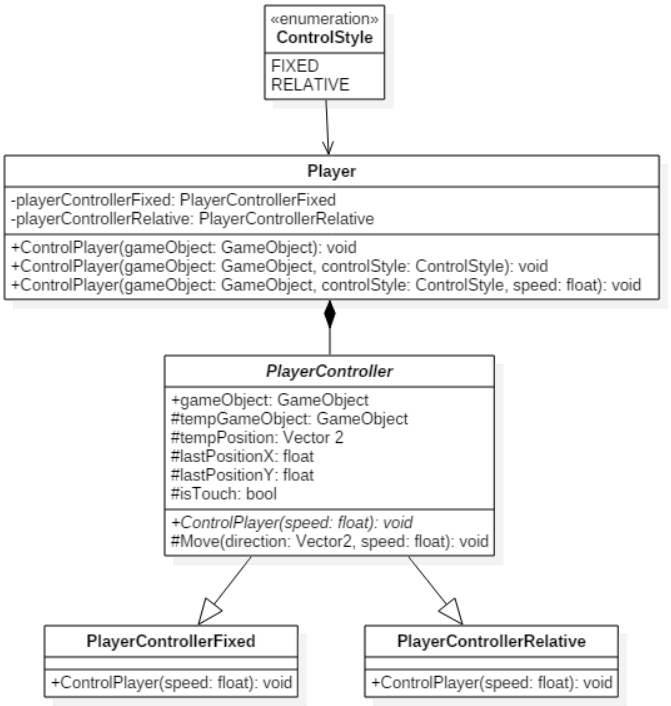

Fig. 4. Class diagram hotspot character control.

3.5. Implementation class library

Class Library will be implemented to C# (C Sharp) language [11]. Naming identifier standards such as properties, classes, and methods at the implementation stage will be adjusted to the C# language naming rules [12].

3.6. Unit testing

Unit Testing is done by creating a test case method for each class Scrolling Shooter library function [13]. The case test method is based on the unit test scenario's design, which can be seen in Table 3.

The unit test results show that all Scrolling Shooter Class Library methods can function properly according to their functions.

Table 3. Test case scenario.

Class Test	Method Test	Status
Player	ControlPlayer()	Success
Background	ScrollBG()	Success
Spawner	Spawn()	Success
Data	Save()	Success
	Load()	Success
Destroyer	Destroy()	Success
DisplayUI	Display()	Success
Movement	Move()	Success
Shoot	Shoot()	Success
Player	ControlPlayer()	Success

3.7. Integration testing

After unit testing is complete, the next testing stage is integration testing [14, 15]. Integration testing is done by directly testing each unit in the class library to create a scrolling shooter game prototype. In this study, integration testing was tested directly by experienced programmers in building games using Unity [16]. The display of the game prototype from the results of integration testing can be seen in Fig. 5.

Fig. 5. Prototype game scrolling shooter.

4. Conclusion

Based on the results of integration testing, it is concluded that programmers can build scrolling shooter games without having to make the entire system functional from scratch using the class library. It is proven that the class library built in this study can produce a scrolling shooter game prototype consisting of a vertical scrolling shooter game and a horizontal scrolling shooter game. It is because the class library that has been built has provided basic functions that can be reused by programmers to build scrolling shooter games.

References

1. Atanasova, N.; Todorovski, L.; Džeroski, S.; and Kompare, B. (2006). Constructing a library of domain knowledge for automated modelling of aquatic ecosystems. *Ecological Modelling*, 194(1-3), 14-36.
2. Hofschuster, W.; and Krämer, W. (2012). *C-xsc: a C++ class library for extended scientific computing*. New York: Springer, Berlin.

3. Atmaja, P.W.; and Mandyartha, E.P. (2020). Difficulty curve-based procedural generation of scrolling shooter enemy formations. *Proceedings of the International Conference Science and Technology.* Surabaya, Indonesia, 1569, 1-8.
4. Seaman, C.B. (1999). Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, 25(4), 557-572.
5. Kienle, H.M.; and Müller, H.A. (2010). Rigi—an environment for software reverse engineering, exploration, visualization, and redocumentation. *Science of Computer Programming*, 75(4), 247-263.
6. Mowlaei, S.; Schmidt, S.; Zadtootaghaj, S.; and Möller, S. (2018). Know your game: a bottom-up approach for gaming research. *Proceedings of the Tenth International Conference on Quality of Multimedia Experience*. Cagliari, Italian,. 1-3.
7. Ştefan, A.; Stănescu, I.A.; and Hauge, J.M.B. (2016). Approaches to reengineering digital games. *Proceedings of the International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Charlotte, North Carolina, USA.
8. Pree, W. (2000). Hot-spot-driven framework development. *Framework*, 2(B1), 1-13.
9. Fayad, M.E.; Schmidt, D.C.; and Johnson, R.E. (1999). *Building application frameworks: object-oriented foundations of framework design*. New York*:* John Wiley & Sons Inc.
10. Forina, M.; Oliveri, P.; Lanteri, S.; and Casale, M. (2008). Class-modelling techniques, classic and new, for old and new problems. *Chemometrics and Intelligent Laboratory Systems*, 93(2), 132-148.
11. Kokholm, N.; and Sestoft, P. (2006). The c5 generic collection library for c# and cli. the IT university of Copenhagen. Retrieved February 15, 2021, from https://www.itu.dk/research/c5/latest/ITU-TR-2006-76.pdf.
12. Schuller, D. (2011). *C# game programming: for serious game creation* (1st ed.). United States: Course Technology.
13. Sawant, A.A.; Bari, P.H.; and Chawan, P.M. (2012). Software testing techniques and strategies Software testing techniques and strategies. *International Journal of Engineering Research and Applications*, 2(3), 980-986.
14. Bissi, W.; Neto, A.G.S.S.; and Emer, M.C.F.P. (2016). The effects of test driven development on internal quality, external quality and productivity: a systematic review. *Information and Software Technology*, 74, 45-54.
15. Ali, S.; Briand, L.C.; Rehman, M.J.; Asghar, H.; Iqbal, M.Z.Z.; and Nadeem, A. (2007). A state-based approach to integration testing based on UML models. *Information and Software Technology*, 49(11-12), 1087-1106.
16. Wazir, H.K.; and Annaz, F.Y. (2014). Using unity for 3D object orientation in a virtual environment. *Proceedings of the 5th Brunei International Conference on Engineering and Technology.* Bandar Seri Begawan, Brunei, 1-6.