UDC 004.4

OBJECT-ORIENTED PROGRAMMING PLAYS A CRUCIAL ROLE IN GAME DEVELOPMENT

Author – Roman Molchanov[1], Stud. of gr. KS-21
Scientific supervisor – Yevhen Plakhtii[2], Senior Lecturer
Language consultant – Anastasiia Plakhtii[3], Cand. Sc. (Philol.), Assoc. Prof.
[1]romch3dmv@gmail.com, [2]plakhtii.ev@gmail.com,
[3]plakhtiy.anastasiya@pdaba.edu.ua
Prydniprovska State Academy of Civil Engineering and Architecture

Object-oriented programming (OOP) plays a crucial role in game development, as it allows developers to create complex systems and mechanics that are easy to understand, maintain and extend.

When it comes to game development, it is important to understand that it is not just about programming, it is about creating a whole system with a great idea that needs to be engaging, challenging, and perfect in every way. To achieve this goal, game developers use a variety of tools and technologies, including OOP [1].

OOP is a programming methodology that breaks down complex systems into a multitude of simpler, interacting objects. This allows developers to create more understandable, flexible, and maintainable code and also facilitates the process of adding new objects, new features and mechanics to the game. OOP is a fundamental element of many popular game engines, such as Unity and Unreal Engine.

In this work, we explore why OOP is critically important in game development, as well as provide a number of specific examples of OOP usage in the game industry. We will also examine potential issues and limitations associated with using OOP in game development.

One of the main advantages of OOP in game development is its ability to break down game code into separate modules and classes, making maintenance and development easier. Each module can contain its own data and methods that can be protected from changes by other parts of the program. This reduces the possibility of errors and improves overall application security.

OOP also provides inheritance, where new classes can inherit properties and methods from a parent class. This reduces code duplication and speeds up the development of new features and game elements. In addition, polymorphism, which is a part of OOP, allows working with objects of different classes but with the same interface, making coding much easier and increasing the flexibility of the application.

Game development requires a significant amount of code and resources, and OOP can greatly reduce the time and resources spent on development. Developers can create more unified and structured code that can be reused in other games and projects. Furthermore, by structuring the code using OOP, support and development of the application become easier and more efficient.

Examples of OOP in game development include the use of classes and objects to create game characters, items and the game world, as well as to manage the game process and logic [2].

One example of OOP in game development is the use of classes to create game characters. Each character can be represented as an object of a class containing information about the character's attributes such as health, speed, and strength [2]. The class can also contain methods to control the character's behavior in the game, such as movement through the game world or attacking enemies.

Another example of OOP in game development is the use of classes to create game items such as weapons, armor and potions. Each item can be represented as an object of a class containing information about its properties such as damage, defense, and potion effects. The class can also contain methods for using and manipulating the item in the game.

One of example of OOP in game development is the use of classes to create the game world. Each element of the game world, such as a building, plant, or landscape, can be represented as an object of a class containing information about the element's properties such as size, shape and texture. The class can also contain methods to control the element's behavior in the game, such as displaying and animating it.

Finally, OOP can be used to manage the game process and logic. For example, classes can be used to create game states such as start, pause and end. Classes can also be used to manage events in the game such as defeating a boss or achieving a goal.

All of these examples demonstrate how OOP can be used in game development to create more convenient, flexible and understandable code that can be reused in different projects.

Although OOP is widely used in game development, there are several issues associated with its use:

- Code redundancy. The use of classes and objects can lead to an increase in code volume, making it more difficult to read and maintain.
- Performance. Creating a large number of objects can slow down game performance, especially on devices with limited resources.
- Complexity. Developing with OOP can be more challenging than writing procedural code, especially for novice developers.

- Inheritance. While inheritance is one of the main principles of OOP, incorrect or overused can result in the creation of complex and hard-to-understand class structures.
- Rigidity. The use of OOP can lead to the creation of structures that are too inflexible, making it difficult to modify code when necessary.
- Debugging difficulties. Dividing code into multiple classes can make it difficult to find errors and debug the program, especially if errors occur at the boundaries of classes and objects [3].

In conclusion, it can be noted that OOP is a widely used approach in game development, which allows creating more convenient, flexible, and understandable code that can be reused in various projects. However, its use can lead to several problems, such as code redundancy, performance issues, complexity, inheritance problems, inflexibility and debugging difficulties. These issues should be taken into account when choosing an approach to game development, and their importance for a particular project should be evaluated. However, when used correctly, OOP can significantly ease game development and make it more efficient and convenient for developers and users. Ultimately, we conclude that OOP is a necessary tool for creating games.

References

1. Vujošević-Janičić M. and Tošić D. The role of programming paradigms in the first programming courses. The Teaching of Mathematics. 2008, no. 21, pp. 63−83.

2. Ivanova S. Learning computer programming through games development. Conference proceedings of eLearning and Software for Education (eLSE). Carol I National Defence University Publishing House, 2016, vol. 12, no. 01, pp. 492−497.

3. Brown N.C.C. and Wilson G. Ten quick tips for teaching programming. PLoS computational biology. 2018, vol. 14, no. 4, p. e1006023. URL: https://doi.org/10.1371/journal.pcbi.1006023.