Development of algorithm for continuous generation of a computer game in terms of usability and optimization of developed code in computer science

Tibor Skala[1], Vladimir Cviljušac[1], Kristijan Mustač[1]

[1]University of Zagreb, Faculty of Graphic Arts, Getaldićeva 2, Zagreb, Croatia
E-mail: tibor.skala@grf.hr, vladimir1.cviljusac@grf.hr, kristijan.mustac@gmail.com

Abstract

As both hardware and software have become increasingly available and constantly developed, they globally contribute to improvements in technology in every field of technology and arts. Digital tools for creation and processing of graphical contents are very developed and they have been designed to shorten the time required for content creation, which is, in this case, animation. Since contemporary animation has experienced a surge in various visual styles and visualization methods, programming is built-in in everything that is currently in use. There is no doubt that there is a variety of algorithms and software which are the brain and the moving force behind any idea created for a specific purpose and applicability in society. Art and technology combined make a direct and oriented medium for publishing and marketing in every industry, including those which are not necessarily closely related to those that rely heavily on visual aspect of work. Additionally, quality and consistency of an algorithm will also depend on proper integration into the system that will be powered by that algorithm as well as on the way the algorithm is designed. Development of an endless algorithm and its effective use will be shown during the use of the computer game. In order to present the effect of various parameters, in the final phase of the computer game development an endless algorithm was tested with varying number of key input parameters (achieved time, score reached, pace of the game).

Keywords: object-oriented programming, computer game, endless algorithm

1. Introduction

Developing of an endless algorithm was devised and tested through its implementation in a computer game. Adobe Flash Builder software was used, as it was designed as a software used to compress ideas from concept into working, i.e. programming part. A satisfactory ratio of gameplay was achieved by testing and subsequent balancing of various parameters.

2. Flash Builder

Adobe Flash Builder is an integrated development environment (IDE) for development of cross-platforms, rich Internet applications for desktop computers and a growing number of mobile devices [1]. Development environment of Flash Builder features testing and debugging functionalities as well as profiling tools, which contribute to enhanced and effective production.

Figure 1. Flash Builder GUI

Flash Builder is developed in Eclipse (open source IDE). It is a full-featured package, containing all the tools required for development of applications that rely on open code, such as Flex framework and ActionScript 3.0 language, which are also used by this game based on endless algorithm.

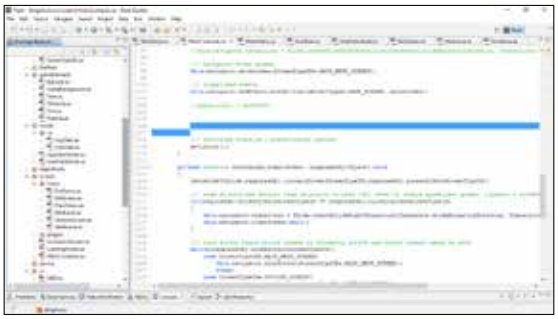

Figure 2 – in-game screen classes

An infinite loop was created in the game. As the name suggests, the loop repeats one action recurrently. The purpose of endless algorithm is to achieve balance of the game according to the player's abilities and base parameters of the game [2].

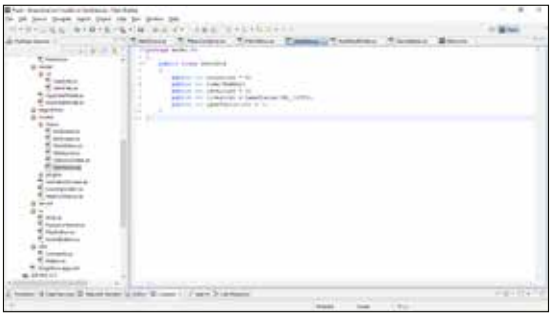

Figure 3 – base parameters of the game

After a level has been completed, a new level is run. The end results of the previously completed level are the input parameters of a new level. Functions that are run when the player has completed one game level are presented in Figure 4. After a successful completion of a level, the player proceeds to next level, carrying the sum of lives and points accumulated at the moment of completion of the previous one. If the player fails to complete a level, she/he is automatically demoted to the initial level with the initial set of parameters, such as number of lives and points (Figure 3).

Figure 4 – function executed at the moment of level completion

The portion of the programme code that is employed to balance the game is in fact an infinite loop and that is the core portion of the game that makes the game interesting and endless. The advantage of such an approach is that player can play the game for as long as she/he wishes so as the algorithm is responsible to maintain the player's interest at high level.

Figure 5 – functions used to store the player's parameters

In order for the loop to be able to be run within the video game, a code has been written and assigned to each screen (screens are presented on Figure 2). It is the very core that forms the endless algorithm. Figure 6 depicts the flow of the screens in an infinite loop. The initial screen is shown. It is followed by a screen selected according to a key pressed by the player.

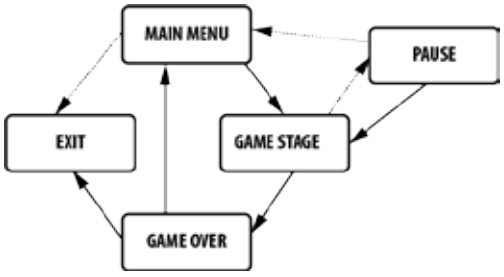

Figure 6 – illustration of endless algorithm

On the initial screen (Figure 7), every graphic object is initialized, background is cropped to the screen size and the base game parameters are set (points, lives etc.). The figure also shows the functions that produce the parameters associated with the player, based on the success or failure in the previously played level.

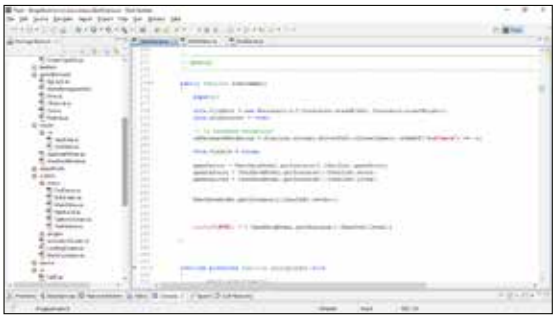

Figure 7 – code listing of the initial screen

Figure 8 – code listing of the screen navigation section

To each screen (Figure 2) within the game certain syntax, i.e. defining code, has been assigned [3]. There is a specific piece of code and certain settings associated with each screen. The structure of the settings defined in the code can be shown through simplifying the game architecture.

"// go to the main menu screen
// press start button to initiate the game elements (player, score, time, lives, enemies etc.)
// when you start you have a default save state (0 points - 3 lives – default timer = 1 min)
// when you reach the end of the level with a certain sum of points (your score) and lives you given a choice:
a) go to the next level with current lives and points
b) restart the level and start from 0 points and 3 lives
c) go to the main menu
// if you don't reach the end of the level (you lose the last life) you have a choice:
b) restart the level and start from 0 points and 3 lives
c) go to the main menu"

However, there is one setting introduced to produce the mentioned endless algorithm. Code listing presented on the Figure 8 is used for specific navigation between the screens (user's instructions determine the navigation directly). The part of the code where the endless algorithm is achieved lies in there and a simplified game architecture is shown in the figure below (Figure 9) [4].

The section of the code for screen navigation is placed in the "MainContainer" class (Figure 10).

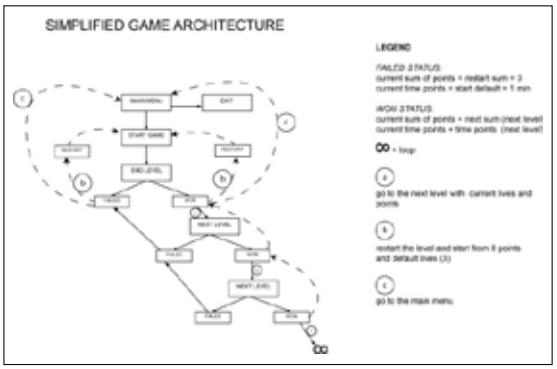

Figure 9 – simplified game architecture

As its name suggests, it acts as a main container for the functions on which the algorithm relies [5], [6]. Also, managing the parameters from one central point is much more useful for achieving the desired effect than using more functions placed in more sections for the same operation [7], [8], [9]. A unique identification code is assigned to each screen. It is used by the "switch" logical function to change the identification code of the used screen at that particular moment [10]. The change of the code also affects the theme of the used screen (main menu, splash screen, pause etc.).

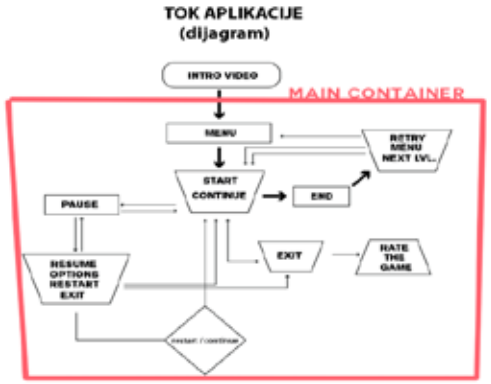

Figure 10 – flow diagram of a computer game

Depending on the user's input, the course of the game follows the flow diagram in Figure 10. The part of the code highlighted in red rectangle contains parts of the code listing responsible for navigation through all the screens except the intro video. The main, i.e. central container for the functions is within that rectangle.

3. Conclusion

The advantage of use of such algorithm is primarily in more efficient use and optimization of the used programme. In this particular case, it is a computer game [11]. As the given examples show, by employing such algorithm the player can play the game for as long as he/she wishes to, while the algorithm is responsible to keep the player constantly interested in playing and does so by recurring increases or decreases of the game settings [12]. As it was already mentioned above, the effect of the code in the main function container is to ensure that each screen that presents a certain theme (main menu, pause, exit, etc.) also records the user's current results. The results may later be either kept or reset [13], [14]. Also, managing the parameters from one central point is much more convenient for achieving the intended effect of cohesion and the feel of central management as opposed to the method of using more various functions for the same operation. It is precisely for that reason that this game consumes less processing and graphic resources and consequently it also ends faster.

In the computer games field of the computer science it is imperative to provide entertainment for the users and users require new challenges in prolonged engagement in the game. Therefore it is vital for the computer game to be able to efficiently provide content that will maintain high level of user's enthusiasm for the game. This can be achieved precisely by using such an algorithm. Additionally, there is a number of other fields in which it can also be put to use [15]. Such algorithm can also be used to improve specific segments in other types of applications. Uses include easier and more precise communication of storage and upgrade of settings. The results may later be either kept or reset.

An example of such non-specific use of these kind of algorithm within an application could be its use in infinite loop or running a programme that would continuously record data on passengers in traffic (time, station, ID of the route, alternative routes and similar) [16].

This algorithm can certainly be widely used in fields of technology awareness, but in this case the main issue would be its capabilities of being upgraded. With significantly increased number of screens and available options for upgrades within the same computer game, an important requirement would be to introduce a set of parallel junction points within the code, which would be able to simultaneously control the increased number of screens and data storages.

4. Reference

1 ***http://www.adobe.com/products/flash-builder.html
2 Mustač K. (2016), Razvoj algoritma za generiranje beskonačne računalne igre s GPU
3 ***http://computer.howstuffworks.com/question717.htm
4 ***http://whatis.techtarget.com/definition/algorithm
5 ***http://algs4.cs.princeton.edu/11model/
6 ***http://docbook.rasip.fer.hr/ddb/res/35/Ch4.html
7 ***https://www.khanacademy.org/computing/computer-science/algorithms
8 ***http://study.com/academy/lesson/what-is-an-algorithm-in-programming-definition-examples-analysis.html
9 ***http://adrianmejia.com/blog/2014/02/13/algorithms-for-dummies-part-1-sorting/
10 Sanjay Madhav, Game Programming Algorithms and Techniques: A Platform-Agnostic Approach, Addison-Wesley Professional, 2014.
11 Jialin Liu, Olivier Teytaud,Tristan Cazenave, Fast Seed-Learning Algorithms for Games, Computers and Games: 9th International Conference, CG 2016, Leiden, The Netherlands, Springer 2016 edition, 2017.
12 Saint-Pierre, D.L., Teytaud, O.: Nash and the bandit approach for adversarial portfolios. In: CIG 2014 - Computational Intelligence in Games, pp. 1–7. IEEE, Dortmund, August 2014.
13 Gaudel, R., Hoock, J.B., Pérez, J., Sokolovska, N., Teytaud, O.: A principled method for exploiting opening books. In: International Conference on Computers and Games, pp. 136–144, Kanazawa, Japan, 2010.
14 Méhat, J., Cazenave, T.: A parallel general game player. KI-Künstliche Intell. 25, 43–47, 2011.
15 Mohamed, T.P., Hruschka, E.R.J., Mitchell, T.M.: Discovering relations between noun categories. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1447–1455, 2011.
16 Alena Kozlova, Joseph Alexander Brown and Elizabeth Reading, 2015 IEEE Conference on Computational Intelligence and Games, Examination of Representational Expression in Maze Generation Algorithms.