Journal of information and communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 10(2): 168-174, Jun. 2012 **Regular Paper**

Design of an Alpine Skiing Game Using ActionScript 3.0

Bai-Tiantain[1], Jong-Hoon Park[1*], and Chul-Won Kim[2], *Member, KIICE*

[1]Department of Computer, Joongbu University, Geumsan 312-702, Korea
[2]Department of Computer Engineering, Honam University, Gwangju 506-714, Korea

Abstract

Flash is the most popular 2D animation and game development software, supporting vector and media technology at the core, which enables the development of small and pithy games. It is widely used in Web animation, courseware, TV commercials, game development, and other works of design. In this paper, we propose a control movement function and auxiliary functions for an alpine skiing game based on ActionScript 3.0. The control movement function is designed with moving phases (i.e., free fall, sliding, projectile, and landing). The auxiliary functions include drawing path, start/restart movement, and saving the highest score. In addition, for the visual design of our game, we designed animations in connection with a character and background. In order to facilitate testing the friction, users can input a chosen friction value. Without user input, the friction default is set at 0.97.

Index Terms: Flash, Game, ActionScript 3.0, Moving phases

I. INTRODUCTION

ActionScript 3.0 is a powerful, object-oriented programming language that signifies an important step in the evolution of the capabilities of the Flash Player runtime. The motivation driving ActionScript 3.0 is to create a language ideally suited for rapidly building rich internet applications, which have become an essential part of the web experience [1, 2]. ActionScript 3.0 is based on ECMAScript, the international standardized programming language for scripting. ActionScript 3.0 is compliant with the ECMAScript language specification, third edition. It also contains functionality based on ongoing work on ECMAScript edition 4, occurring within the ECMA standards body [3].

ActionScript is executed by the ActionScript virtual machine (AVM) built into the Flash Player. AVM1, the virtual machine used to execute legacy ActionScript code, powers the Flash Player today and makes possible a wide range of interactive media and rich internet applications. There are numerous products that generate content and applications targeted at the Flash Player runtime [4]. Often, these products incorporate support for ActionScript to add interactivity and behavior to their output. In the Adobe product family, professional designers and developers might use ActionScript within several tools and servers - such as Flash, Flex, and Flash Media Server - to create content and applications for the Flash Player. The Flex product family, including the new Eclipse-based Flex Builder 2 IDE, will be the first product line to access the new capabilities of ActionScript 3.0.

This paper designs an alpine skiing game by means of ActionScript 3.0. We propose a control movement function and auxiliary functions for game controls. The control movement is composed of moving objects (i.e., Santa Claus). It proceeds through four different phases of motion:

Received 06 March 2012, **Revised** 30 March 2012, **Accepted** 09 April 2012
***Corresponding Author** E-mail: jhpark@joongbu.ac.kr

Open Access **http://dx.doi.org/10.6109/jicce.2012.10.2.168** print ISSN:2234-8255 online ISSN:2234-8883

1) free fall, 2) sliding, 3) projectile, and 4) landing. In free fall, the object has uniformly accelerated motion before touching the path. Sliding is the object's motion when it touches the path. Projectile motion happens when the object leaves the path of movement. Finally, landing, in which the object lands on the ground to stop movement, is last. The auxiliary functions include drawing the path, starting/ restarting movement, and saving the highest score. Furthermore, to maximize the visual design of our game, animations of a character and background were created. Finally, the moving objects have friction when they slide on the path of the ski slope. In order to facilitate testing the friction, users can input the friction by themselves. If a user does not input any data, the friction will default to 0.97.

This paper is organized as follows: section 2 reviews the relational technologies of the game; section 3 describes the idea of the game's design; section 4 presents the design and implementation of the proposed game. Finally, section 5 concludes this paper.

II. RELATIONAL TECHNOLOGIES OF THE GAME

A. ActionScript 3.0

In this paper, we use Flash CS4 as a tool for developing the alpine skiing game. Flash CS4 is the newest of the Flash authoring tools and is compatible with ActionScript 3.0. Besides, it is a good tool for creating games [1-4].

ActionScript 3.0 is designed to address the following goals.

- Safety: the language supports type safety so developers can write unambiguous, easily maintainable code.
- Simplicity: the language is intuitive enough for developers to be able to read and write programs without constantly consulting a reference manual.
- Performance: the language enables developers to write complex programs that perform efficiently and responsively.
- Compatibility: the language provides a short backward and forward compatibility path and a significant overlap with industry standards. ActionScript 3.0 is a dialect of ECMAScript that formalizes the features of ActionScript 2.0, adds the capabilities of ECMAScript for extensible markup language (XML; E4X), and unifies the language into a coherent whole.

B. Previous Games

1) Alpine Skiing Game by Armor Games Company

The game is produced by Armor games in Fig. 1. This game, called Alpine skiing, uses the arrow keys to control the object motion [5]. It is according to different path to the decision speed. It also applies many formulas of physics, such as those for free fall and horizontal projectile motion. Thus this game has a very high indicative.

Fig. 1. Armor's alpine skiing game.

2) Snowboard Game by GameBetty

The game is produced by Game Betty in Fig. 2. This game is calls Snowboard and uses the mouse to control the direction [6]. The game speed is fixed, so there is no physics formula to determine the speed and direction. Only by moving the mouse can the object be controlled.

Fig. 2. Game Betty's snowboard game.

3) Speed Sled Game by Wedu

The game shown in Fig. 3, called Speed Sled, is produced by a company called Wedu. It involves clicking on the point by the upper left corner to calculate the initial velocity [7]. This game path is determined; it can only be determined based on the initial velocity of the displacement. Click the button at the top left corner to control the initial velocity. If the button is pressed for a longer time, the initial velocity will be faster, and if the button is pressed for a shorter time, then the initial velocity will be slower.

Fig. 3. Wedu's speed sled game.

III. DESIGN OF ALPINE SKIING GAME

A. Design Ideas

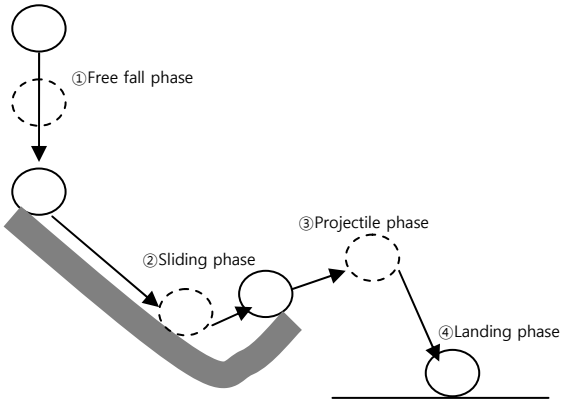

Fig. 4. The four different phases of motion.

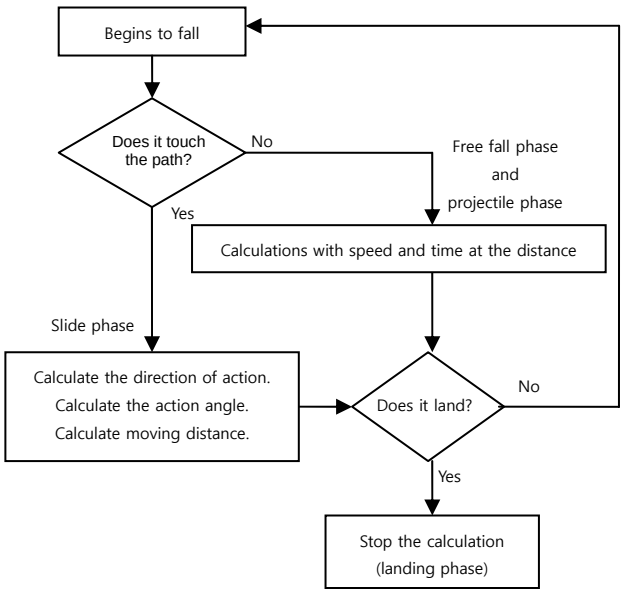

Fig. 5. The control process of object motion.

In this section the design of the moving objects (i.e., Santa Claus) is described, specifically the process of moving through four different phases of motion as follows: free fall, sliding, projectile motion, and landing. In Fig. 4, the free fall is the state in which the object is in uniformly accelerated motion before touching the path. Sliding is the object's motion when it touches the path. Projectile motion describes the behavior of the object when it leaves the path. Landing is the last stage, in which the object lands on the ground and its movement ends.

The control of the movement function is the core of the game. We also designed other auxiliary functions. These functions include drawing the path, starting/restarting movement and saving the highest score. To make the game more aesthetically pleasing, other animation was also needed. For example, in this game, the object is to have friction when it is sliding on the path. The object operation flow is shown in Fig. 5.

B. Application of the Equation Algorithm

The key part of the game is the difficulty of programming; these difficulties include the displacement calculation, the direction determination, and angle calculations in a variety of motion states.

1) Free Fall and Projectile Motion Phases

The free fall and projectile motion phases can be calculated using the following formula [8]:

$$dvyDown = \text{Movement direction} \times \text{Initial velocity} \times \sin(\text{Angle}) + g \times t \quad (1)$$

$$dvxDown = \text{Movement direction} \times \text{Initial velocity} \times \cos(\text{Angle}) \quad (2)$$

The free fall phase is the object's first movement phase. The "dvyDown" and "dvxDown" are the vertical and horizontal movement of the displacement components. "Movement direction" has an initial value of 0. That is free fall. When the object touches the path, the direction of object movement is determined according to the angle of the path. Movement of the object to the right has a value of 1. Movement of the object to the left has a value of -1.

The projectile phase is the object's third movement phase after the sliding phase. "Initial velocity" is the motion state of the object at that time. The timer t is 0. The initial angle is Math.PI/2. That is free fall, straight down. When the object touches the path, we can calculate the angle according to the path. The angle range is (-Math.PI, +Math.PI). When the object goes out of the path, its speed will remain on the same angle and perform projectile motion.

2) Sliding Phase

The sliding phase is the object's second movement phase after the free fall phase. As the path is drawn by the user, the angle is not fixed. Therefore, when the object is sliding on the path, the movement direction is also not fixed. We can use the following formula to calculate the displacement of the object [9].

dvyDown = Movement direction × Initial velocity × sin(Angle) (3)

dvxDown = Movement direction × Initial velocity × cos(Angle) (4)

Here the "Initial velocity" is the speed of the previous value after exercise; the "Angle" is based on the calculated path.

3) Calculation of the Friction

Friction is a force of reverse direction against speed. After setting a friction value, we can subtract it from the speed. In fact, it is subtracted from the speed itself and not from the x-axis or y-axis. If it is from the x-axis and y-axis, the object will keep along the constant angle. If one of the speeds reaches 0 early, the object will continue its vertical or horizontal movement for a while. Therefore, we find the angular velocity according to the speed and direction. The velocity vector can then be subtracted from the speed. If the friction is greater than the speed, the speed becomes 0. The formula is as follows [10]:

vx = Math.cos(Angle) × (Speed-friction) (5)

vy = Math.sin(Angle) × (Speed-friction) (6)

In order to facilitate testing the friction, in this game, users can input the friction by themselves. If the user does not input any data, the friction will default to 0.97.

IV. IMPLEMENTATION OF ALPINE SKIING GAME

A. Implementation of Alpine Skiing Game

This is a small Flash game; after running the game, users can draw a white glide path directly on the stage and click the play button. Command on the hilltop Santa Claus sliding down the drive along the path. The movement of Santa Claus will be based on the flight distance, and the slope of the length of the path varies. The game will record the highest score for each slide. In this game, users can input the friction themselves. Fig. 6 demonstrates the effect of the film running time [11].

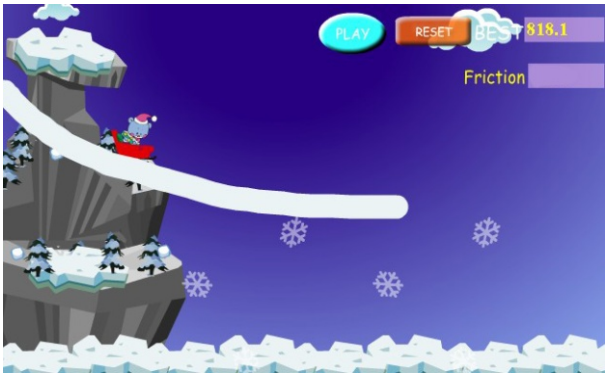

Fig. 6. Screenshot of alpine skiing game.

By the preceding analysis, the components required in the film can be initially determined, in addition to the background elements and some other modifications of the animation. The main elements include moving objects (Santa Claus on skis), the scoreboard, the start-up and restore movement for the two buttons, and a record display object.

B. Running Screenshot

1) Free Fall Phase

Before the Santa Claus touches the line, it will free-fall, and there is no lateral movement. This is shown in Fig. 7.

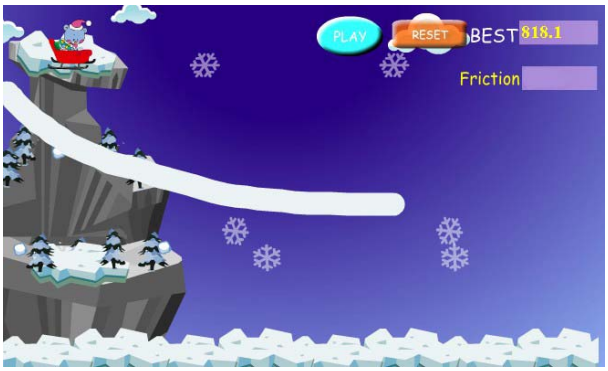

Fig. 7. Free fall phase.

2) Sliding Phase

When Santa Claus touches the line, it starts sliding along the line and there it has lateral displacement, as shown in Fig. 8.

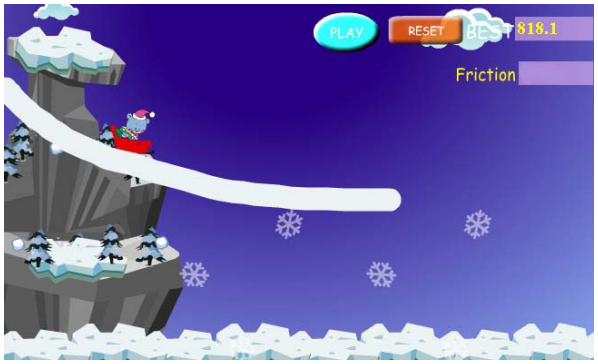

Fig. 8. Sliding phase.

3) Projectile Phase

The Santa Claus figure leaves the line according to the initial velocity, direction and angle of projectile motion, as shown in Fig. 9.

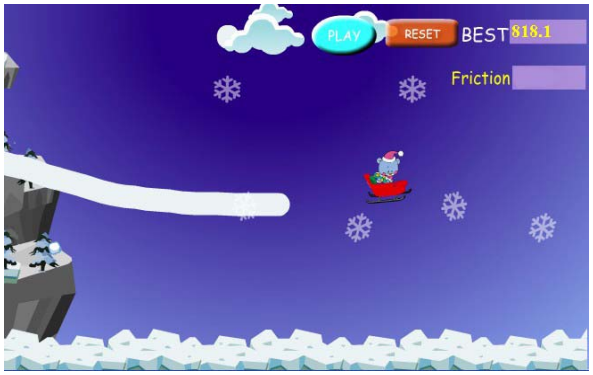

Fig. 9. Projectile phase.

4) Landing Phase and Stop

When Santa Claus lands, the object stops motion and the x-axis is calculated. The scoreboard appears with scores. This is shown in Fig. 10.

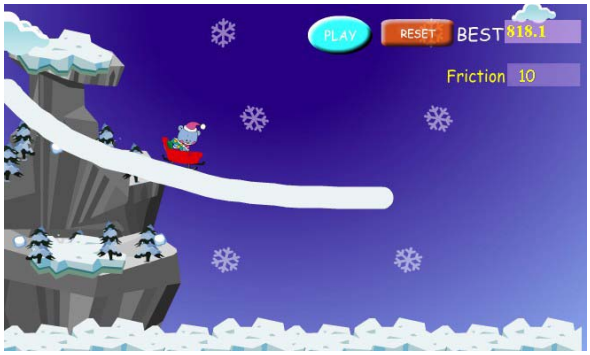

Fig. 10. Landing phase

5) Friction Test

In the input box on the right, the user can adjust the friction force. Fig. 11 shows a friction value (2) and the distance of motion is shorter than Fig. 10 (default value = 0.97). As Fig. 12 shows, the object stops on the path because of excessive friction.

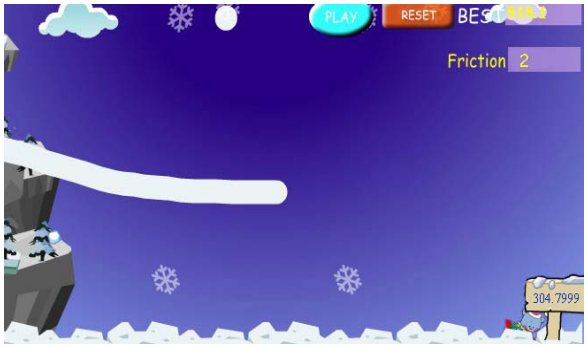

Fig. 11. Input of friction value (2)

Fig. 12. Input of Friction value (10).

C. Comparison and Consideration

The following Table 1 shows the comparison of other games.

The game is still not perfect. As a game, it is too simple. It is difficult for players to have sustained interest in it. Its playability is not excellent, nor are the visuals of high enough quality. The game needs improvement before commercialization.

On the other hand, this game has realistic movement. In addition to the common physics formulas, the friction formula was included. Users can input the friction themselves. This makes the movement more realistic. Realistic object movement is advantage strength of the game. In the game, the user can draw a custom path. This makes the object movement more diverse, and is another advantage of the game.

Given the core formula and design ideas, it is hoped that the game can be successfully commercialized.

Table 1. Comparison of other games

Game's name	How to calculate motion	Path	Friction
Alpine skiing game in this paper	Direction (decided by user) Angle (decided by user) Initial velocity (decided by user)	Drawn by user	Reflected (decided by user)
Alpine skiing game by the Armor Games	Direction (fixed) Angle (fixed) Initial velocity (fixed)	Fixed path	Not reflected
Snowboard game by the GameBetty	Fixed	Fixed path	Not reflected
Speed sled game by Wedu	Direction (fixed) Angle (fixed) Initial velocity (decided by user)	Fixed path	Not reflected

V. CONCLUSIONS

In this paper, we designed a skiing game which includes moving objects, movement controls, and auxiliary functions. The movement control functions consist of free fall, sliding, projectile motion, and landing. The auxiliary functions include drawing a path, starting/restarting movement, and saving the highest score. In addition, to create an aesthetically pleasing visual design for our game, we designed animations with a character and background. Finally, the object experiences friction when it slides on the path. In order to facilitate testing the friction, users can input the friction themselves. If the user does not input any data, the friction will default to 0.97. This game needs improvement because it is too simple, and it is difficult for players sustain interest in the game. However, this game has realistic movement. Therefore, this game's function can be used in physics simulation and testing. In addition, it can improve other games by using the developed models.

REFERENCES

[1] Z. G. Zhu, L. Mou, and Y. L. Chen, *Flash ActionScript 3.0 Programming Tutorials*, Beijing: Tsinghua University Press, 2008.

[2] G. Grossman and E. Huang, ActionScript 3.0 overview [Internet]. Available: http://www.adobe.com/devnet/actionscript/articles/actionscript3_overview.html.

[3] S. Yung, *Flash ActionScript 3 Palace Road*, Beijing: Electronic Industry Press, 2007.

[4] X. Yin, *Flash 8 ActionScript Interactive Effects Design of 108 Cases*, Beijing: China Youth Publishing House, 2006.

[5] 17173.com, Alpine skiing game by amore games [Internet]. http://flash.17173.com/flashfile/2011-03-29/20110329110933874.shtml.

[6] 17173.com, Snowboard Betty game by gameshero [Internet]. http://flash.17173.com/flashfile/2007-12-11/20071211105216987.shtml.

[7] 17173.com, Wedu toboggan jump 2001 game [Internet]. http://flash.17173.com/flashfile/2006-07-09/20060709175546195.shtml.

[8] C. Mook, *Essential ActionScript 3.0*, Sebastopol, CA: O'Reilly, 2007.

[9] Z. P. Li and Y. D. Rong, *Flash CS4 Animation Project Training Tutorial*, Beijing: Jiaotong University Press, 2010.

[10] X. B. Li and F. Zhou, *Flash CS4 Animation Design Collection Case*, Beijing: China Water Resources and Hydropower Press, 2010.

[11] Y. H. Zhu and M. Tian, *Flash CS4 and Case-Based Tutoria*l, Beijing: Mechanical Industry Press, 2010.

Bai Tiantian
received the M.S. degree from the Department of Information Science from Joongbu University, Chungnam, Korea in 2012. Her research interests include game simulation and game engines.

Jong-Hoon Park
received the M.S. and Ph.D. degrees from the Department of Computer Engineering of Kwangwoon University, Seoul, Korea in 1987 and 1995, respectively. From 1995 to 1998, he worked at the National Computerization Agency. Since 1999, he has been a professor in the Dept. of Computer Science, University of Joongbu, Korea. His research interests include internet search, the semantic Web, and game engines.

Chul-Won Kim
received the Ph.D. degree in Computer Engeineering,kwangwoon University in 1997. He is a Professor at Honam University. His research interests include Image Processing, Multimedia Information Retrieval, and Multimedia Processing.