Journal of The Korea Society of Computer and Information
Vol. 23 No. 9, pp. 21-26, September 2018
www.ksci.re.kr
https://doi.org/10.9708/jksci.2018.23.09.021

Design of Boat Racing Game using Buoyant Force

Syoungog An*, Jae-Hyun Kang*, Soo Kyun Kim*

Abstract

A variety of game algorithms have been proposed in recent years. Racing games using these new algorithms have greatly improved compared to those in the past. According to the analysis of racing games done by wheelgamer.com, a website that specializes in racing games, cars are the most common subject in mainstream racing games. These results show there are very few racing games using special subjects such as boats. This paper proposes a method to develop a realistic boat racing game using the Buoyont Force, beyond the limitation of typical racing games that are restricted to cars. A Material Editor based on Unreal Engine 4.12 is used to manifest the Dynamic Material of realistic ocean water. Blueprint, a visual scripting function, is used to materialize buoyancy which allows the boat to move freely along the waves.

▸Keyword: Buoyant Force, Dynamic, Blueprint, Racing Game, Game Engine

I. Introduction

Gran Track 10 is known to be the first racing game, launched in 1974 as an Atari model. In the 1980s and 90s, there was not enough hardware to express the speed of racing. Racing games could only be played in arcades. Breakthroughs in game algorithms and hardware allowed racing games to make a huge leap forward. The graphics became more realistic of course, but so did the handling. The invention of the specialized controller allowed for the player to feel like he/she was actually driving. Also, the ability to select and tune the vehicles is a characteristic exclusive to racing games. According to the statistics of the racing game website wheelgamer.com1, 86.3% of racing games use cars as the main subject whereas only 1% use special subjects such as boats.

The game proposed in this study is a boat racing game that goes beyond the limitations of typical racing games. The contrast of the ocean water graphics is adjusted using the Material Editor3-5 of the Unreal Engine2. Dynamic Material is used to manifest realistic wave motions. Buoyancy is materialized using a visual scripting function called Blue Print6 , allowing the boat to float in response to the wave motion. The boat's steering algorithm is applied here, allowing the player to move the boat in the desired direction and speed. Simple interface operation is implemented by using Adobe PhotoShop CS5 and an Unreal UI editing tool called UMG. Also, various boat characters are modeled with 3Ds Max, adding to the game's appeal.

• First Author: Syoungog An, Corresponding Author: Soo Kyun Kim
*Syoungog An (sungohk@pcu.ac.kr), Dept. of Game Engineering, Pai Chai University
*Jae-Hyun Kang (minlog@empal.com), Dept. of Game Engineering, Pai Chai University
*Soo Kyun Kim (kimsk@korea.ac.kr), Dept. of Game Engineering, Pai Chai University
• Received: 2018. 09. 11, Revised: 2018. 09. 15, Accepted: 2018. 09. 17.
• This work was supported by the research grant of Pai Chai University in 2018.

II. Design of Game Structure

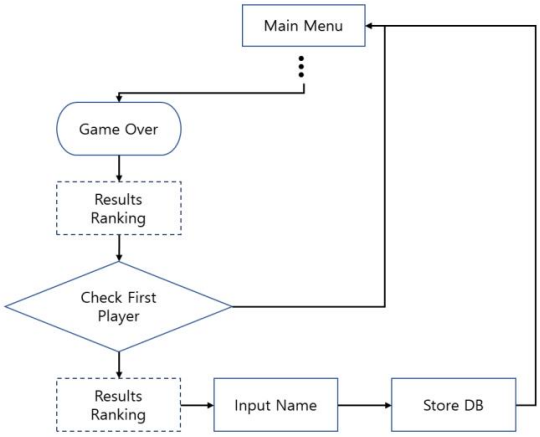

Fig. 1. Flow chart of main menu

The game proposed in this paper is a first person boat racing game. In order to win, the player must race three laps around the track and arrive at the finish line before all seven AI boats. The game UI is designed to adjust the direction of the boat using the arrow keys on the keyboard, the drift function using the shift key, and the speed booster function using the control key.

Fig. 1 is a flowchart of the main menu. The starting screen of the proposed game is the main menu screen. When the Single Player button is selected, the UI screen for selecting the boat and the map is displayed. After the selection process a loading screen is displayed, then switched to the game scene. The player will now wait for a three second countdown before starting the race.

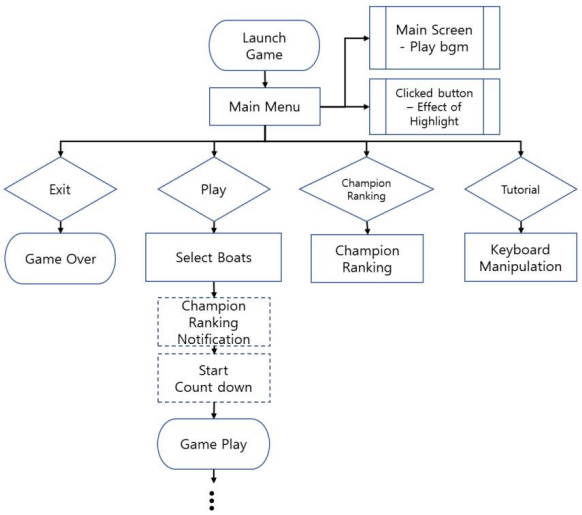

Fig. 2. Flow chart of game scene

Fig. 2 shows the flow chart of the game scene. The player races with seven AI boats. The boat's movement is determined by the player's input key. Buoyancy and ocean water graphics are updated in live time by detecting any collisions on the boat. One lap is counted every time the boat passes through the finish line, and when the count reaches three laps the game is over.

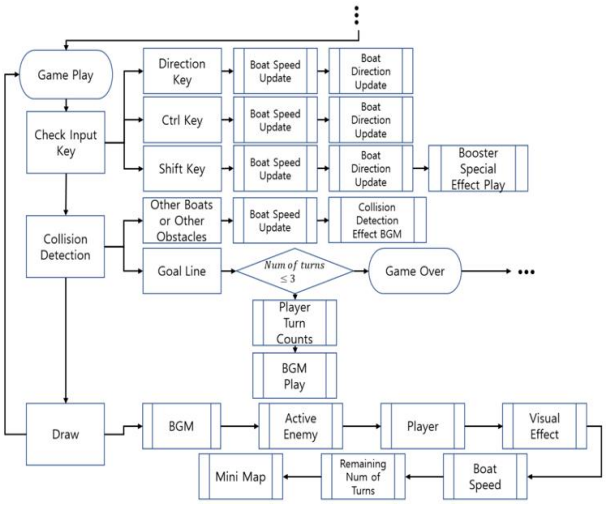

Fig. 3. Flow chart of game over

Fig. 3 shows the flow chart until the game is over. At the end of the game, the results ranking window will be displayed briefly before returning to the main screen. If the player reached the finish line first before all other AIs, the player's name and the race record will be stored in the champion rankings and can be retrieved from the main screen at any time.

III. Buoyancy Calculations and Boat Design

1. Graphic Representation of Ocean Water

The ocean water graphics used in this game is based on the 'Ocean Water Shader'[7], which is shared in the Unreal Community Project, and is materialized using the Material Editor. First, the parameters for the Base Color of the ocean water are set as Dark Color and Light Color. Then these values are manipulated in order to express diversely colorful ocean water that differs from scene to scene. Also, the color and depth of the wave foams is adjusted to make the movement of the boat upon on the ocean water look more natural. Lastly, Translucent Material is used to control the transparency of the ocean water and make it look clearer when the player is near

the shore and underwater Fig. 4 shows the results of the Ocean Water.

Fig. 4. Ocean graphic of main menu scene

2. Representation of Buoyancy

As shown in Fig. 5, buoyancy8 is caused by opposing pressures within fluid. The total sum of the horizontal forces is zero, but the total sum of the vertical forces equals the force against the bottom of the boat minus the force against the top of the boat. The force (f) equals the pressure (P) multiplied by the area (A). (ρ: Density)

Force

$f_t = p_t A_t = pgh_t s^2$

$f_b = p_b A_b = pgh_b s^2$

Differential

$F_b - f_t = pgh_b s^2 - pgh_t s^2$

$= pg(h_b - gh_t)s^2$

$= -pgs^3$

$= -pgV$ (V: Volume)

Fig. 5. Enemy Character

The specific gravity an object and the magnitude of buoyancy interact with each other based on Archimedes' law8-9. An object with lower specific gravity is influenced more by buoyancy rather than weight. When the volume of the object submerged is the same, buoyancy is greater if the specific gravity of the fluid is greater.

1. Density
   - > 1 … sink
   - < 1 … float
2. Force Analysis
   - → weight vs. buoyant force
   - weight > buoyant force = sink
   - weight < buoyant force = float

Fig. 6. Two ways of floating in water

The buoyancy formula is used to create a blueprint of the boat's interface. Fig. 6 shows the results of this blueprint. First, a virtual cuboid called the box collision is created at the bottom of the boat's static mesh. The bottom of the box collision becomes the actual surface area that floats in the ocean. Then, virtual physics objects called the buoyancy force are installed to check the position and angle on the map that changes in accordance to the surface area and movement of the box collision. As a result, the blueprint variables (height of the wave, velocity, water volume, etc.) that manage the dynamic material in the ocean water are updated in real time. Also, the algorithm shown in Fig. 6 is processed as a Construct Script in order to execute it into the most basic script of the boat within the game.

3. Materialization of Boat Movement

Axis Mapping list is set up in the project settings in order to bind the player's input key and the boat's movement. Set the Up, Down, Left, Right keys of the keyboard to MoveForward, MoveRight. The values of the scale should be set as + 1.0 for the up, right key and -1.0 for the down, left key.

Then retrieve the preset MoveForward, MoveRight list from the boat's blueprint and set the Custom Event to Run on Server so that it only moves when the player is holding down the keys, and check Reliable. An arrow component called the steer force location is installed in the front bottom surface of the boat facing forward in order to command the direction of the boat. This serves as the steering wheel for the boat, and the direction is determined by keyboard input.

The initial values of the float variables forward speed and steering speed should be set as 1500.0 and 200.0, respectively. The change in direction is determined by multiplying the key input scale values of 1 and -1 to these variables. Float variables called the drift value and boost value are created in order to create drift and booster skills, with an initial value of 1.0. Each time the shift and ctrl keys are pressed this value is increased to 2.0 for three seconds to materialize a skill event that speeds up the redirection and forwards speeds. Fig. 7 below shows the pseudo codes for the boat's movement loop.

1. **Axis Mappings**:
   MoveForward (Keyboard ↑, ↓ / scale 1.0, -1.0)
   MoveRight (Keyboard →, ← / scale 1.0, -1.0)
   Arrow Component: Steer Force Location
   Float Variables:
   Steering Speed (200.0), Forward Speed (1500.0)
   Right Axis Value (0.0), Forward Axis Value (0.0)
   Drift Value (1.0), Boost Value(1.0)
2. **for** t ← **to** ∞ **do**
   t ← delta seconds
   Event 1:
   Add Force at Location ← Target = boat static mesh
   ← Location = Steer Force Location of Value of World Location
   ← Force = Steer Force Location of Value of World Rotation
   Right Vector * (Mass of Boat * Steering Speed
   * Right Axis Value * Drift Value)

   Event 2:
   Add Force ← Target = boat static mesh
   ← Force = Forward Vector of Boat * (Mass of
   Boat * Forward Speed * Forward Axis Value *
   Drift Value)
3. **if** MoveRight **then** Right Axis Value = 1.0 or –1.0
   then Add Force at Location
   Execution of Event
4. **if** MoveForward **then** Forward Axis Value = 1.0 or -1.0
   then Add Force
   Execution of Event
5. **if** Wave Value of Ocean material ← 300 to 700
   then Z Value of Steer Force Location ← 150 to 200

Fig. 7. Pseudo code for applying movement forward/right forces

IV. Design of User Interface

1. Main Menu Screen

The first screen of the proposed game is the main menu screen, which consists of Single Player, Champion Ranking, Help & Options, and Exit game, as shown in Fig. 8.

Fig. 8. Main menu scene

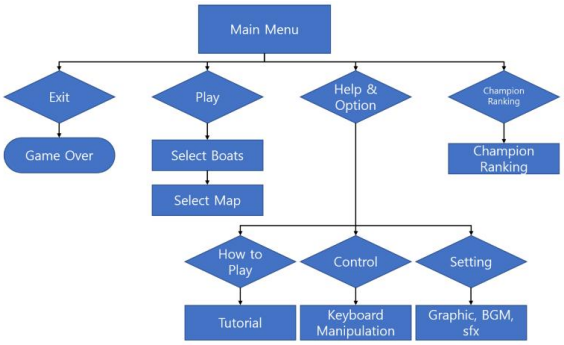

Fig. 9. Flow chart of main menu

Fig. 9 shows a flowchart of the contents in each selection of the main menu.

2. Creation of Various Menus

After pressing the Single Player button, the game is started immediately after selecting the boat and map. Fig. 10 shows the boat selection menu.

Fig. 10. Selection of boat menu

Fig. 11. Selection of map menu

Fig. 11 shows the map selection menu, where the player can select the racing track.

The Champion Ranking menu allows the player to check the names and race records of all previous players who came in first place. The Help & Options menu shows the How to Play, Controls, and Settings buttons. These options can teach the player how to play the game and

how to operate the keyboard, and also allows the setting for graphics, BGM, and SFX.

V. Development Environment

The proposed game in this study is a first person boat racing game developed with Unreal Engine 4.12 and operated in Windows platform. Two maps called the Atlantis and Brown City were designed for the boat race. The UI is updated in real time while the game is operated. The time and the number of laps around the track are shown in the upper left corner of the screen. Real-time race ranking is displayed in the upper right Position of the screen. The player is able to see his/her current position by using the mini map on the bottom of the screen. The speedometer and drift gauge can be used to check the speed of the boat and whether or not the booster is being used. Fig. 12 shows a game scene from the Atlantis map.

Fig. 12. Game scene of atlantis map

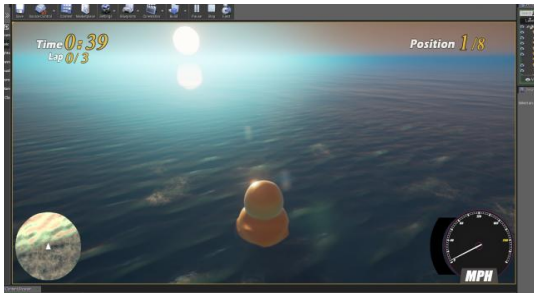

Fig. 13. Game scene of brown city map

Fig. 13 shows a game scene from the Brown City map.

VI. Conclusions

The game proposed in this study is a racing simulation game that overcomes the limitation of typical racing games and uses boats rather than cars. Some strong points include the ability to control the contrast of the ocean water graphics using Material Editor of the Unreal Engine, and to express realistic wave motion by implementing dynamic material. A distinct feature of the game is the Blue Print, a visual scripting function, which is used to materialize buoyancy in order to allow the boat to float in response to wave motion. Also, the steering algorithm of the boat allows the player to move the boat in the desired direction and speed.

REFERENCES

[1] Wheelgamer, Article(CrossRef Link)
[2] Unreal Engine, Article(CrossRef Link)
[3] Aram Cookson, “Unreal Engine 4 GameDevelopment in 24 Hours, Sams Teach Yourself”, Sams Publishing; 1 edition, June 18, (2016)
[4] Tom Shannon, “Unreal Engine 4 for Design Visualization: Developing Stunning Interactive Visualizations, Animations, and Renderings”, Addison - Wesley Professional; 1edition, August 2017
[5] Joanna Lee, “Learning Unreal Engine Game Development”, Packt Publishing - ebooks Account, February 2016
[6] Brenden Sewell , “Blueprints Visual Scripting for Unreal Engine”, Packt Publishing - ebooks Account, July 2015
[7] David Nixon , “Unreal Engine 4 for Beginners”, Luquinox, February 2017
[8] Khan Academy, Article(CrossRef Link)
[9] WIKIPEDIA- Buoyancy, Article(CrossRef Link)
[10] WIKIPEDIA- Archimedes, Article(CrossRef Link)
[11] The Story Of Mathmeatics, Article(CrossRef Link)

Authors

Syungog An received the Ph.D. degrees in Computer Science and Engineering from Korea University, Korea, in 1989. She is currently a Professor in the Department of Game Engineering, Paichai University.

Jae-Hyun Kang received the B.S. degrees in GameEngineering from Paichai University, Korea,

Soo Kyun Kim received Ph.D. in Computer Science & Engineering Department of Korea University, Seoul, Korea, in 2006. He joined Telecommunication R&D center at Samsung Electronics Co., Ltd., from 2006 and 2008. He is now a professor at Department of Game Engineering at Paichai University, Korea. His research interests include multimedia, pattern recognition, image processing, mobile graphics, geometric modeling, and interactive computer graphics. He is a member of ACM, IEEE, IEEE CS, KACE, KMMS, KKITS and KIIT.