International Journal of Advanced Science and Technology
Vol.95 (2016), pp.73-88
http://dx.doi.org/10.14257/ijast.2016.95.07

Design and Implementation of a Three-Dimensional Game Based on a Brain-Computer Interface

Han-Joong Kang[1], Dong-Hyun Kim[1], Hyoung-Cheol Lee, Byeong Man Kim[1], Dukhwan Oh[1] and Sung-Bong Jang[1,*]

[1]Department of Computer Software Engineering, Kumoh National Institute of Technology, 61 Daehak-ro, Gumi, Kyoung-Buk, 730-701, Republic of Korea
palbong70@gmail.com

Abstract

This paper describes design and implementation of a three-dimensional game based on a brain-computer interface (BCI). The interaction between player and game application is controlled by translating brain signals into user-key input through electroencephalogram software. To evaluate the game performance, we compare the scores of the BCI interface with those of an existing non-BCI interface. The experiments results show that the players can control the game with BCI although the game speed is low.

Keywords: *Brain-Computer Interface (BCI), 3D game*

1. Introduction

The brain-computer interface is a new interface technology that controls a computer using only the signals of a human brainwave (electroencephalogram, EEG) without any physical interaction between the human and the computer; it is so called BCI technology [01]. Though BCI technology has been traditionally used to help the physically disabled, it has recently attracted attention in diverse fields such as games [02], augmented reality experiences [03], and eye tracking.

The basic principle of the game is to capture diverse electric signals emitted from the game player's brain using an EEG device, and deliver the signals to the game software to enable the game to progress. EEG (electroencephalography) can be defined as the electric activities of the brain generated from the scalp [04].

Human brainwaves show a pattern of complicated waveforms. When brainwaves are analyzed, they are first converted into frequency bands. These frequency bands show diverse distributions from low to high frequencies, and can be generally divided into Delta (1.0–3.5 Hz), Theta (4.0–7.5 Hz), Alpha (8.0–13.0 Hz), Beta (14.0–30.0 Hz), and Gamma (30.0–100.0 Hz) waves. Among these, Alpha, Beta, and Gamma waves are generated in relative quantity from the frontal lobe and the parietal lobe, when a person is mentally and physically stabilized, anxious or nervous, or when extremely aroused or excited, respectively. In addition, Delta and Theta waves are generated when a person sleeps.

In this study, a 3D game that is controlled by the player through a brainwave-based interface is implemented, and its performance is evaluated.

This paper is organized as follows; Section 2 discusses about the related works; Section 3 presents a 3D game system based on BCI interface. Section 4 describes experimental results. Finally, section 5 concludes the paper.

* Corresponding Author

ISSN: 2005-4238 IJAST

2. Related Works

Brainwave-based games have been attempted many times. The first one was attempted by Vidal [04] in 1977; it is a game where the user solves a labyrinth. The game player is asked to focus their eyes or posture on four fixed points outside the screen. The game moves in the direction of the fixed point where the user focuses; *i.e.*, the movement in the labyrinth is controlled by predicting the neural activities by vision. The four fixed points, arranged in the form of a diamond, stimulate the neural activities of the brain visual cortex by having a light periodically blink among the four fixed points. Even though this was the first attempt for a BCI game, its performance was very good. An easier method of applying brainwave signals to a game is to use the frequencies, such as Alpha, Beta, or Gamma, coming out of the human brain. This method was first attempted in a game called "Brainball," which was developed by (Hjelm) [05]. Two players play the game wearing EEG headsets. The headset measures the brainwaves of each player and sends them to the computer. There, each player's relaxation score is calculated by measuring the distribution ratio of the Alpha and Beta waves sent from the EEG. A more relaxed person produces a higher score. The ball is moved across a table depending on this score. The player who moves the ball to the end of the table most quickly wins the game. One interesting note is that, when a ball has almost reached the end of the table, the player sometimes becomes excited at the thought of winning, which may cause the score to drop, due to excess Beta-wave emissions, and cause the player to lose the game. Van der Laar (2013) [06] applied a BCI interface to the online role-playing game "World of Warcraft" (WoW). This game is a role-playing game played by multiple players online. The objective is to raise the character's level and obtain better abilities and weapons by accomplishing quests, killing enemies, exploring the world, and gaining experience. Van der Laar developed a BCI version of this game using only Alpha waves. In the BCI version of this game, the game character can change from one form to another as the user moves between a stable state and an aroused state (the state where the Alpha-wave activity is increased). The night-elf Druid form, which heavily depends on intellectual power and concentration, is related to a stabilized arousal state. A decrease in the Alpha-wave activity, generally caused by stress or anxiety, makes the night elf change into the form of a bear (which means a fight is wanted). The users can train themselves to use their Alpha waves very quickly. The authors say that the users have learnt how to control Alpha waves such that they can intentionally transform with the Alpha waves every five seconds.

Another example of utilizing a BCI game is in neuro-feedback (training to self control brainwaves). This was used to treat children with Attention Deficit Hyperactivity Disorder (ADHD) in an experiment carried out by Pope and Palsson [07]. The children played a Sony PlayStation video game. One group used standard neuro-feedback as the control-device input, while the other used a neuro-feedback system developed by NASA (National Aeronautics and Space Administration). The latter, more-sensitive control device showed an accurate brainwave pattern. Learning to control through brain signals is the goal of the game in such an application. The peculiarity of a neuro-feedback game is that the player must learn to control a specific part of the brain to proficiently play the game. It is expected that such learning can be utilized in real life.

Motor control based on BCI is regarded as a traditional input device in BCI games, in contrast to the previous method. For example, in the first-person 3D shooting game developed by Pineda in 2003, rather than controlling the forward and backward movements using physical buttons, the movements were controlled using the mu rhythm of the motor cortex. No mechanical learning was involved and the players learnt how to control the mu rhythm by training themselves for 10 hours over a five-week course.

Another motor-control BCI game is Pac-Man, developed by Krepkl in 2007. In this game, movements are detected using lateralized readiness potential (LRP) signals (spikes in the brain's electrical activities that take place when a person's arm, leg, or foot goes into a movable state). This is a slow negative change that occurs occasionally in the brainwave signal of an activated motor cortex before a motion is actually started. In the BCI game, Pac-Man takes a step every 1.5 to 2 seconds, and moves straight ahead until it receives a command to turn or touches a wall. The users said that they felt Pac-Man moving in the correct direction before they consciously decided on it. This is an example of a new level of interaction mechanism. An interaction that takes place unconsciously in such a natural manner can only be possible in a BCI game.

3. Design and Implementation of 3D BCI Game System

3.1. Design of BCI Interface for the Game

The Emotiv EPOC Control Panel program was used as the interface between the game and the brainwaves. While games require many signals, brainwave signals are rather limited. It is somewhat difficult to make diverse brainwaves determine mouse coordinates and the cognition ranges of keyboard buttons. However, signal processing can be used to convert the waves into a state where the signals can be classified and accepted to some extent. The signals are classified using the method each developer defines.

In this study, brainwaves are converted into general mouse and keyboard signals through the Emotiv EPOC Control Panel program. The game reflects when the mouse coordinates are used and the keys of a keyboard are pressed.

The Emotiv EPOC Control Panel program measures the brainwaves it receives from the user wearing the brainwave monitor, and converts the result into the desired computer signals. When the Emotiv EPOC Control Panel first receives the brainwave signals, it converts them into basic program signals using predefined brainwave signals, and shows the processing results using images in the program. It can also convert the desired motions into keyboard input signals through a separate setting. Motions that are not processed into key input signals after registering the desired motions in the program can be converted through the following steps.

step 1) Measurement of the basic state,

step 2) Measurement of the image that pushes an object, and

step 3) Measurement of the image that pulls an object.

Mouse signals can also be determined using brainwave signals. However, in this case, the mouse cursor movements are not handled through brainwaves; the data is input through a sensor in the brainwave monitor itself. This study uses mouse-cursor movements acquired through the brainwave monitor, and key inputs made by the user's image.

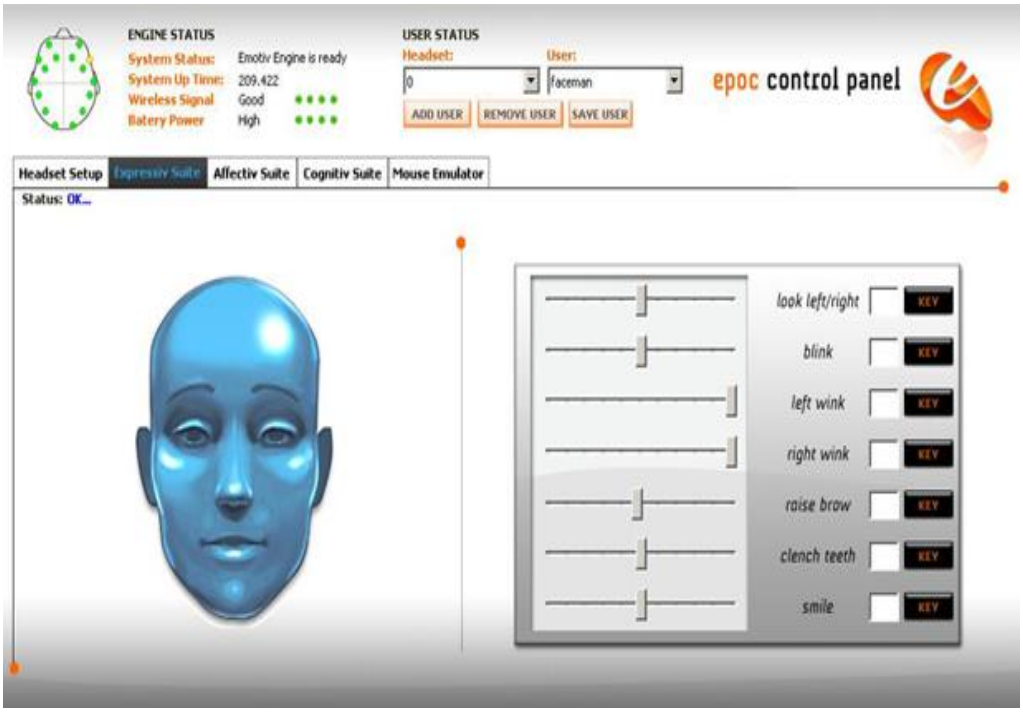

Figure 1. Keys Definition of the BCI Interaction in Emotiv EPOC

3.2. Design of Game Application

The game scenario used in this study is as follows: A spacecraft suddenly invades the earth and destroys the land. After several months, it settles down on earth and periodically attacks the surrounding area. The spacecraft is not easily accessible as its surroundings are enclosed within a powerful protective shield. Accordingly, the humans build an unmanned control robot to break into the spacecraft through an underground tunnel.

The player (robot) who enters the spacecraft fights with the boss, and eventually succeeds in defeating it. Once the player has escaped from the spacecraft, he should eliminate the fragments to protect the surrounding area. The composition of each scene, based on the above scenario, is shown in Table 1.

The game based on the above scenario was implemented using integrated authoring tool Unity3D version 5.0, which is used to produce interactive content such as 3D video games, architecture visualization, and real-time 3D animations. The game scenes were organized using Unity3D and saved as separate files. The events required for the scenes and the handler to process them were implemented using JavaScript and C#. A JavaScript file or a C# file was applied to each GameObject in a Scene to control the GameObject's actions. Unity3D was used to organize and build the entire form of the game.

Table 1. Description of Game Processing Scenes

Game Start	When the game-start button is pressed, the Scene switches to cinema1.
Cinema1	The alien invasion is shown to explain why the player breaks into the spacecraft.
Cinema2	This scene shows the player entering the underground tunnel. An enemy is discovered and the entry battle is announced.
Entry Battle	The monster attacks the player. Points can be acquired by deflecting the monster's attack, and points are reduced if the player's attack misses.
Cinema3	This scene shows escaping the underground tunnel and sneaking into the spacecraft. The player is caught and forcefully pulled into the spacecraft by the boss. The battle against the boss begins immediately.
Battle against Boss	The boss's sphere attacks the player. If the player's shot goes astray, it is placed under the control of the boss and fired back at the player. If the boss is disposed of, the scene moves to an escape battle.
Escape Battle	The player completes the mission and runs away from the spacecraft. To prevent the fragments of the spacecraft from destroying the surrounding area, the fragments are shot and eliminated.
Ending	The score is output along with the ending credits, and the score is saved in the rank file.

A Scene refers to the file used to organize a scene in Unity3D; it stores the current state of the GameObjects arranged by the developer; loading a Scene file in the Unity3D program also retrieves the previously worked content. GameObjects refer to objects in the scene, which were implemented using the C# and JavaScript languages. Figure 2 shows examples of the simple JavaScript and C# code used in this paper.

```
#pragma strict
var speed=100;
function Start () {
  yield WaitForSeconds(1);
  Destroy(gameObject);
  score.ScoreDown();
}
function Update () {
   var amtToMove=speed*Time.deltaTime;
   transform.Translate(Vector3.forward*amtToMove);
}

function OnCollisionEnter(coll:Collision) {
   Debug.Log(coll.gameObject.tag);
   if(coll.gameObject.tag=="MonsterAttack") {
                    score.ScoreUp();
   } else           {
     score.ScoreDown();
   }
  Destroy(gameObject);
}
```

```
using UnityEngine;
using System.Collections;
public class EnemyAttack : MonoBehaviour {
  public GameObject missile;
  public GameObject secondMissile;
  public Transform missilePosition;
  public float attackWait;
  public float attackInterval;
  private float second;

  void Start () {   // Use this for initialization
     second = 0;
     StartCoroutine (attackMissile ());
  }
  IEnumerator nextAttackMissile()        {
     StopCoroutine (attackMissile ());
     yield return new WaitForSeconds (3.0f);
  }
}
```

Figure 2. An Example of Implementation using Java and C#

The start screen is divided into difficulty-level selection, game start, and rank checking. The difficulty-level selection is used to designate the game's difficulty level, and includes ordinary and difficult modes. The difficulty level affects the method of acquiring points at

each stage, and the GameObject's events. The game-start function is used to start the game. When the button is clicked by the user, "cinema1" is called. The rank-checking function is used to check the points acquired in the game, and the ranking. When the user clicks the button, the "Ranking" scene appears. The Ranking scene shows the points acquired at the end of the game, along with the name of the player. It also displays the list of players and scores.

A stage is a section where the game can be played and points can be acquired. In this game, three stages were implemented: the entry battle, the battle against the boss, and the escape battle. First, in the entry battle, the player moves forward, eliminating the opposing monster's attacks in a straight-line underground tunnel. The player and the camera move forward toward the exit blocked by the monster. The monster throws a ball at the player, which moves slowly with time. The player is guided by the user's brainwave and shoots a ball toward the mouse pointer. If the player's attack hits the monster's attack, they are both destroyed, and 10 game points are awarded. If the player fails to evade the monster's attack, three game points are deducted. The monster disappears after attacking 15 times. The player then moves to the exit and the scene changes. Second, in the battle against the boss, the player and the boss are located in one space. The player can move as it likes in this space through the input signals. The boss watches the player, and the player can concentrate and deliver an attack in the direction of the mouse pointer, as in the entry battle. The boss fires a floating sphere toward the player at a certain time interval, to interfere with the user's concentration and the player's attack. If the player's attack misses, it is converted into a boss's sphere after a certain period of time. In difficult mode, when the player's attack is converted, three points are deducted. The boss has a weak point on its body; when this weak point is hit by the player, the player receives 10 points and the boss loses health. When the boss's health is fully reduced, it is destroyed, and the scene changes. Third, in the escape battle, the camera is fixed on the burning spacecraft and slowly moves back. A shower of flame pours down from the floating spacecraft periodically, and the player can shoot to hit it. If the player's attack hits a flame, the player gets 10 game points. The player shoots five times in succession every time the user activates the attack. The player's attack can produce a splendid effect by cancelling the flame property, which is difficult when shooting five attacks at once. If the camera moves a certain distance backward, *i.e.*, the player has escaped the battlefield, the scene changes.

4. Performance Evaluation

4.1. System Implementation

Figure 3 shows the start screen of the game.

Figure 3. Configuration of the Initial Start Screen

The GUI software component generates the image texture, game start button, difficulty-level adjustment button, and the view-ranking button. They are displayed on the screen when the game starts. The location and role of each button are adjusted using JavaScript component.

The user changes the scene into the cinema1 scene at the start window. The start window changes the difficulty-level depending on the button pressed by users, and displays the result on the screen. The value of difficulty-level is kept in a JavaScript variable. As the difficulty level variable is static, it affects all parts of the game. This button also generates and stores the score Script.

The view-ranking button calls the rank-checking scene. The rank-checking scene shows the points of the top 10 users. It uses the same GUI as the game start. A button that can be used to return to the game-start scene is generated at the bottom of the rank-checking scene.

The cinema1 scene is implemented through a GameObject shaped like a spacecraft and an ‘unmanned robot’ GameObject is used as the player. The unmanned robot is put inside a small room, which is decorated as a machine laboratory. The GameObject ‘player’ of cinema1 does not have any special motion. Two cameras are used to switch the screen in the cinema image. One shows the spacecraft invading the earth, and the other shows the player’s unmanned control robot. Finally, the Script that generally manages motions is placed in a vacant GameObject to make other GameObjects move in the scene.

In this Script, other GameObjects in the scene are found using the function GameObject.Find(), which is saved in a variable, and the motions are expressed by manipulating the variables to rotate or adjust the position. The spacecraft is made to move forward by adjusting the position, and it drops GameObject ‘bombs’ on the relevant point after slowly stopping. Thus, we show the spacecraft dropping bombs on the earth. In OnGUI(), the explanatory message is printed on the screen over time by updating and measuring the time from OnStart(). Then, the player’s unmanned robot is shown and the cinema2 scene is called.

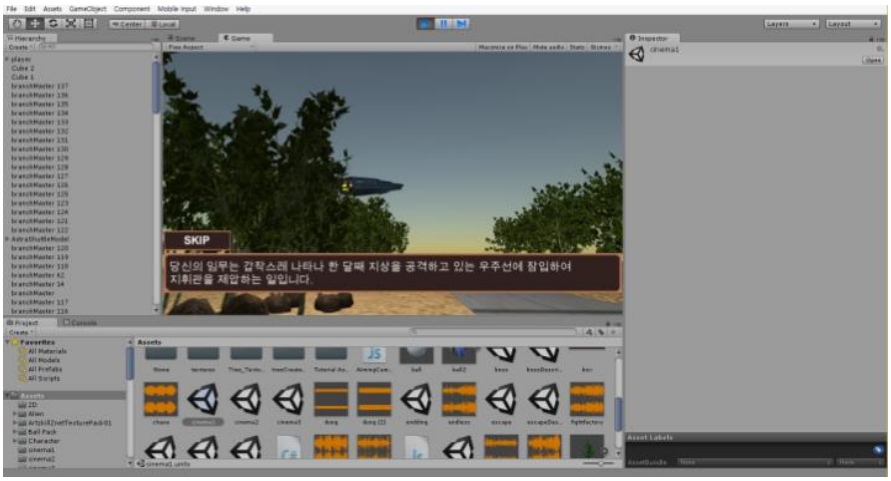

Figure 4. Cinema1 Implementation Screen

In the cinema2 scene, the map to be used in the entry battle is generated, and a GameObject 'monster' is added with the GameObject 'unmanned robot'. The unmanned robot is moved by adding a Script to a vacant GameObject, as in the cinema1 Scene, and the camera is arranged to follow this robot. Whenever the unmanned robot moves to a designated position, an explanatory message is output. The entry battle scene is called after producing the scene where the player encounters the monster. Figure 5 shows the screen of the Cinema2 battle scene.

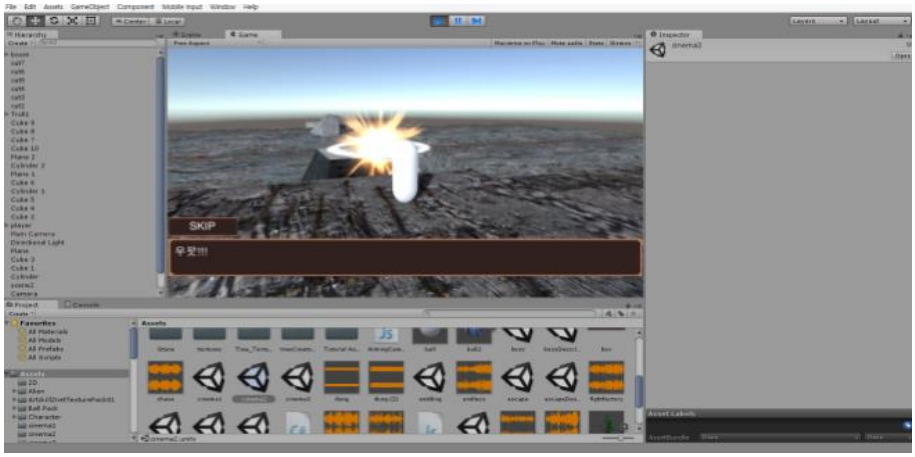

Figure 5. Cinema2 Implementation Screen

In cinema2, the entry battle begins showing the terrain. Points are displayed at the top left of the screen using the GUI, and the position is continuously corrected in the OnUpdate() function so that the unmanned robot slowly approaches the monster. The monster produces a GameObject 'rock' using the Instantiate() function, and throws it at the unmanned robot. If the difficulty level is set to difficult, the monster attacks more frequently. The unmanned robot generates a GameObject 'attack' in front of it, when a signal is received from the user, and shoots it in the direction of the mouse cursor.

The coordinate of the point where the mouse faces is obtained using the Physics.Raycast() function in the OnUpdate() function. The unmanned robot watches this coordinate at all times

using the LookAt() function in the OnUpdate() function. The GameObject 'attack' is shot in the direction where the unmanned robot is looking. The unmanned robot's GameObject 'attack' eliminates itself, using the Destroy() function, when it hits anything. If it hits the monster's GameObject 'attack', the score is increased by calling the score.scoreUp() function. If it hits anywhere else, the score is reduced by calling the score.scoreDown() function.

All the work related to collisions is dealt with by OnCollisionEnter() of the relevant GameObject. The monster calls an animation of killing itself after delivering all of the designated attacks, and disappears through the Destroy() function. Cinema3 is called after the monster's attack is completed and the unmanned robot reaches the designated position.

Figure 6. Entry Battle Implementation Screen

The terrain of the entry battle is used for the cinema3 battle scene, and the GameObjects 'unmanned robot' and 'spacecraft' are generated. The unmanned robot moves toward the spacecraft and an explanatory message is output. When the unmanned robot reaches the spacecraft, a lightning effect is generated and the unmanned robot disappears into the spacecraft. The cinema3 implementation screen is shown in Figure 12.

In the battle against the boss, a huge room appears, along with the 'boss' and 'unmanned robot' GameObjects.

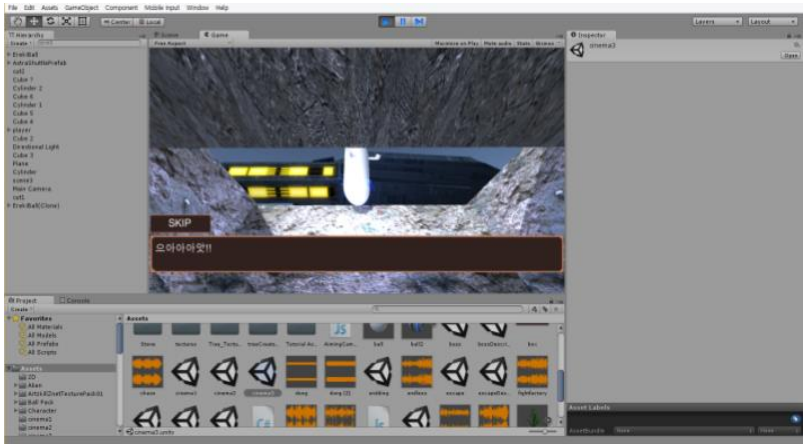

Figure 7. Cinema3 Implementation Screen

The boss's GameObject continuously watches the unmanned robot through the LookAt() function in the OnUpdate() function. The motion of the unmanned robot is controlled by the

signal sent through the character controller Script. The boss's GameObject 'attack' is shot toward the unmanned robot at a designated time interval. The unmanned robot generates its own GameObject 'attack' and shoots it in the direction of the mouse when the user's attack signal is received.

The unmanned robot's attack eliminates itself through the Destroy() function when it collides with the boss's weak point. If it does not strike the boss's weak point within the designated time after being shot, it will change to a boss's attack, eliminating itself after generating a GameObject 'attack' for the boss at its own position. As the boss's GameObject 'attack' already has its own Script, it delivers an attack in the direction of the unmanned robot after it is generated.

A progress bar is placed at the top of the screen to show the boss's health. If the unmanned robot's attack strikes the boss's weak point, the score is raised by calling score.scoreUp() and the boss's health is reduced. If the difficulty level is set to difficult, when an unmanned robot's attack is converted into a boss's attack, score.scoreDown(), which lowers the score, is called. If the boss's health drops below 0, the escape-battle function is called.

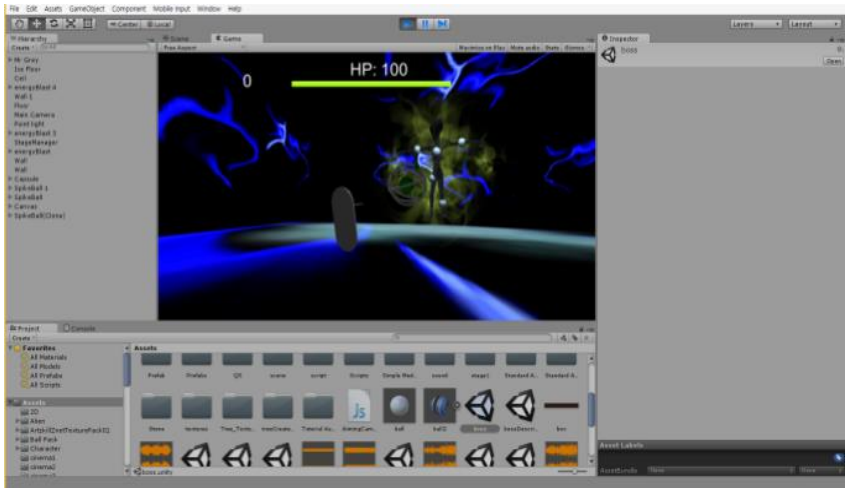

Figure 7. Battle against the Boss Implementation Screen

In the escape battle, the game presents the terrain that expresses the surface of the earth, the GameObject 'unmanned robot', and the GameObject 'spacecraft'. A Script moves the unmanned robot gradually farther back from the spacecraft; the camera also moves backward from the spacecraft, following behind the unmanned robot.

Several fireballs are randomly generated above the spacecraft (using the Random function) after a little time has elapsed. They contain the RigidBody and gravity attributes. These fireballs fall down toward the unmanned robot.

The unmanned robot fires an attack in the direction of the mouse, as described in previous scenes. When it collides with a fireball, the fireball disappears and the score is raised by calling score.scoreUp(). The unmanned robot attacks repeatedly while the input signal continues. If the unmanned robot retreats from the spacecraft to a designated position, the ending scene is called.

The ending credits are displayed using OnGUI. The ending scene randomly moves all the GameObjects used thus far. The ending score is compared with the rank in the file, and then saved at an appropriate position. When the ending credits finish, the game-start scene is called.

4.2. Experiment Result and Analysis

To evaluate our game system's performance, the score obtained using a mouse and a keyboard and the score obtained using brainwaves were comparatively evaluated. Each battle was played eight times. The final scores were recorded for each play.

When the games were played using a mouse and a keyboard, the score was shown to be high, in the range of 900 to 1,300 points. When the BCI interface was used, the score was in the range of 60 to 170 points. As expected, the score of the BCI game was not as good as the existing method. This is believed to be because the mouse pointer and the input timing of the keyboard affected the score.

Figure 7. Brainwave-Interface Game in Progress

A good result could be obtained in the first stage, the entry battle, because the player could easily concentrate. The game rule is simple and there is no surrounding factor for the progress at all. If all 15 balls are hit, 150 points are obtained. When the game is played using a mouse and a keyboard, 144 to 150 points can ordinarily be easily obtained. When playing using brainwaves, although it was not difficult to hit the monster's attacks, many points were lost because the attacks were not shot at the desired time. Between 10 and 90 points were obtained. As brainwaves were used, the error rate depending on the person was shown to be large.

In the second stage, the battle against the boss, the total scores tended to be the same if the difficulty level was ordinary. However, one or two mistakes in difficult mode made the play more difficult, as the score was reduced if an attack failed and the interrupting factors gradually increased. Thus, the gap in the scores tended to be large. This is the stage where concentration is most important in the brainwave game.

Figure.8 Comparison of Game Scores for Brainwave vs. traditional(Mouse/Keyboard) Interface

In the third stage, the escape battle, as points could only be obtained by hitting the fireballs, a very high score could be obtained when a mouse and a keyboard were used. It was easy to hit the fireballs, as the player's balls were continuously shot when the mouse was continually pressed. However, when the brainwaves were used, we could not see whether the balls were continuously shot at this stage. Predicted fire is required to hit the attacks because the flying fireballs are far away at this stage; thus, both timing and aiming ability are considered important. Hitting them using brainwave signals seems to have been a very difficult task. In this stage, 900 points or more could easily be obtained using a mouse and a keyboard; when the brainwave was used, 20 to 80 points could be obtained. Table 2 shows the experiment scores measured at the basic speed.

Table 2. Game Score Comparison for Traditional and BCI Interface at the Basic Speed

No. of Measurements	Game Score , Mouse/Keyboard	Game Score, BCI Interface
1	992	65
2	1004	132
3	1181	85
4	1007	130
5	1001	135
6	1271	122
7	1014	155
8	1201	165
Average	1083.875	123.625

Second, to determine the effect of the game speed on manipulation, an experiment was conducted by increasing the game speed by 20%. The measurement results are shown in Table 3.

Table 3. Game Score with the Game Speed increased by 20%

No. of Measurements	Game Score, Mouse/Keyboard	Game Score, BCI Interface
1	1192	77
2	1294	154
3	1265	179
4	1207	137
5	1316	162
6	1379	147
7	1217	177
8	1312	93
Average	1272.75	140.75

When the resulting scores were analyzed, it could be seen that the scores obtained using the brainwave monitor were as small as 9 to 10% of those with the existing method. When the enemy attack speed was increased by 20%, the score of the existing method decreased by only about 5%, while the BCI game showed a decrease of about 15%.

Finally, when the enemy attack speed was reduced by 20%, the score of the existing method increased by about 20% and that of the BCI game increased by 14%. When the speed of all the processes was reduced by 20%, the score increased in a large scale.

The scores decreased considerably because of the difficulty in controlling the brainwaves. It can be seen that, if the game speed increases, both scores decrease, but the brainwave score decreases further because the brainwave game manipulation is relatively more difficult. If the game speed is reduced, the scores of both methods increase as the game manipulation becomes easier. However, the increase rate of the existing method is higher than that of the brainwave method because its game manipulation is easier. As the game speed has a much bigger effect on the BCI game, we can conclude that neutralizing the difference between the two methods by adjusting the game speed has no effect.

In a general computer game, 36 types of signals are generated using a mouse and a keyboard, if we count only the letters and numbers. This is much larger than the number of signals that can be generated through analytical processing of brainwaves; i.e., the number of signals is insufficient to play an on-line computer game available in the market using brainwaves. However, the game implemented in this study required a smaller number of input signals because it was designed to do manipulation using brainwaves.

Console games use fewer signals; *e.g.*, 8 directions of a joystick or direction keypad, and 6 operation buttons, O, X, ▲, ■, L, and R. Thus, they can more easily be matched to the number of signals that can be generated through brainwave analysis. However, if we simply convert them into games using brainwaves, it can be difficult to enjoy them properly, because

it is rather difficult to generate the brainwave signals at the desired time, as demonstrated in this study.

To enjoy a game, the accuracy of the input signal is essential, and the fact that signals cannot be timely generated is a major disadvantage in the manipulability. It is also difficult as the concentration required to generate brainwaves is distracted by the complexity of the game. Though high concentration can be developed to the extent that the game can be enjoyed using brainwaves, it is somewhat difficult to use it as a simple platform. Who would buy brainwave sensors that are only for a game controlled by brainwaves, or bother to endure such a difficulty to develop the skill to enjoy the game? The answer can be easily determined.

The solution for such a problem is to produce a game taking into account its originality and purpose. The games should be produced with brainwave manipulation in mind, or should be made such that the game is only controlled by brainwaves. In addition, a key goal could be to produce games such that an interesting result is obtained by utilizing the inaccuracy of brainwaves.

For persons who must use brainwaves, the focus is put on patients. There are two possibilities: patients who have a mental problem that needs to be corrected, and patients who have a problem with their fingers. For the former patients, simple concentration programs are already in the market. The latter patients can play the game for fun, just as normal persons do. As a result, useful BCI games can be produced when the emphasis is put on the originality and creativity of the games.

To make an overall conclusion, if a game using brainwaves must be produced, the producer can draw closer to a successful game by producing a system or program that can very precisely receive brainwaves, or by producing original games or games intended to correct concentration.

5. Conclusions

In this study, a game based on brainwave interfaces was designed and implemented, and the performance was evaluated by comparing with the game adopting a keyboard and mouse interface. The results demonstrates that, though it is much more difficult to control this game than general games, it is possible for users to play games with brainwaves interface, and show that the players can control the game with BCI although the game speed is low.

Acknowledgment

This paper was supported by Research Fund, Kumoh National Institute of Technology.

References

[1] T. Wang Vallabhaneni and B. He, “Brain-Computer Interface”, Brain-Computer Interface, **(2005)**, pp. 85-121.
[2] J. C. de Munck, S. I. Gonçalves, R. Mammoliti, R. M. Heethaar and F. H. Lopes da Silva, “Interactions between different EEG frequency bands and their effect on alpha-fMRI correlations”, Neuroimage, vol. 47, no. 1, (**2009**), pp. 69-76.
[3] M.-K. Kim, M. Kim, E. Oh and S.-P. Kim, “A review on the computational methods for emotional state estimation from the human EEG”, Comput. Math. Methods Med., vol. 2013, **(2013)**, pp. 573-734.
[4] F. Lotte, M. Congedo, A. Lécuyer, F. Lamarche and B. Arnaldi, “A review of classification algorithms for EEG-based brain-computer interfaces”, Journal of Neural Eng., vol. 4, no. 2, **(2007)**, pp. R1-R13.
[5] B. Blankertz, G. Curio, K. Müller, N. Group and K. B. Franklin, “Classifying Single Trial EEG: Towards Brain Computer Interfacing”, vol. 10, no. 3, **(2002)**, pp. 210-217.
[6] A. Schlögl, F. Lee, H. Bischof and G. Pfurtscheller, “Characterization of four-class motor imagery EEG data for the BCI-competition 2005”, Journal of Neural Eng., vol. 2, no. 4, **(2005)**, pp. 14-22.

[7] B. Van De Laar, D. O. Bos, B. Reuderink and D. Heylen, "Actual and Imagined Movement in BCI Gaming", vol. 10, no. 5, **(2008)**, pp. 50-65.
[8] D. P. Bos, B. Reuderink, B. Van De Laar, H. Gürkök, C. Mühl, M. Poel, A. Nijholt and D. Heylen, "Brain-Computer Interfacing and Games", **(2010)**, pp. 149-178.
[9] B. van de Laar, H. Gurkok, D. Plass-Oude Bos, M. Poel and A. Nijholt, "Experiencing BCI Control in a Popular Computer Game", IEEE Trans. Comput. Intell. AI Games, vol. 5, no. 2, **(2013)**, pp. 176-184.