Simulated Survey Training Using Game Engine

Moonsik, Kim, South Korea

Key words: education, survey, game engine

Summary

The Game Engines used to develop 3D games has great possibility, so it doesn't stop at making games, it can implement almost everything we can imagine, and we can go inside and try various experiences. The metaverse is an example. We can experience various activities which are impossible in reality through the metaverse. To be a reliable surveyor, it is required to experience various and complex cases. And, of course, you must be familiar with the use of surveying equipment. But, it takes too much time and money. So I thought of an alternative to creating various sample terrain and experiencing various surveys using the game engine and 3D modelling. Fortunately, game engines support accurate location information. And it supports even WGS84. This paper presents a method to develop a survey practice spaces and simulations using a game engine. The game engine provides a high level of immersion with a feeling of being in the real place through 3D graphics processing. And it gives the feeling of using a real equipment through high quality processing of light, shadow, and material. So it provides a much higher level of understanding than just listening to words, reading explanations in books, or looking at pictures. I'll create 3D terrain in Unity 3D and display them on the screen. Using the game engine, we can learn how to survey the various terrains and learn how to use survey equipment as if we are playing games. This method is expected to significantly improve the skill of surveyors.

Simulated Survey Training Using Game Engine

Moonsik, Kim, South Korea

1. Introduction

1.1. 3D Game Engine

Widely known 3D game engines include ones such as Unity and Unreal Engine. Recently, Roblox, a 3D game platform, is gaining its popularity as it newly emerged as the leader in the world of the metaverse. 3D game engines use 3D technology to the limit to show brilliant and amazing special effects and a high level of realism that is indistinguishable from reality. In addition, the recent development of graphic cards and technologies using them has made it possible to process light and shadows that are very similar to real life, resulting in a much higher sense of reality.

Now, 3D game engines are not just for game production, but are being applied in various fields such as architecture, simulation, and education. Since our living space is a three-dimensional world, the target of a 3D game engine can be the entire space we live in. If the training space is 3D modeled and used in the game engine, training will be made possible without the need for expensive educational equipment or space in the real world, and a trainee can redo and repeat desired components. It can bring a greater effect than real-world education where only limited courses are provided.

In addition, it can have positive and immersive educational effect by giving more immediate and impressive feedback with special effects such as fireworks or earning gold whenever you complete a mission. Recently, a lot of effort has been put into gamification, such as making a

game by connecting Go and coding, and it is showing results. Using a 3D game engine for education can be a great help in bringing out high immersion and positive effects.

1.2. Surveyor Education

Since surveying targets a large space, a large space is also required for training with surveying equipment which is costly, which means significant investment needs. Additionally, breakdowns and depreciation are also major problems. This problem is also connected with the difficulty of presenting various survey targets that are essential for training. Engineers who have to work in mountains, fields, and seas need to be exposed to a variety of environments. Even places that are difficult to experience through training, such as places with steep slopes, soft grounds, and places where it is unfit to set up a tripod, should be prepared in various ways and levels before being put into the actual field after studying and experiencing them.

Also, most engineers conduct surveying in the way they are previously trained. Therefore, when faced with the same space and type of survey, each engineer approaches differently. This is where a unified and standardized training method is required to accommodate various surveying types.

Education using a 3D game engine can solve these problems at once. Once created, 3D education program can become a standard on their own. If you create and program a survey process with 3D models of various survey targets just as building a 3D engine video game, you can experience various settings that are difficult to experience in reality at a lower cost.

2. 3D Modeling

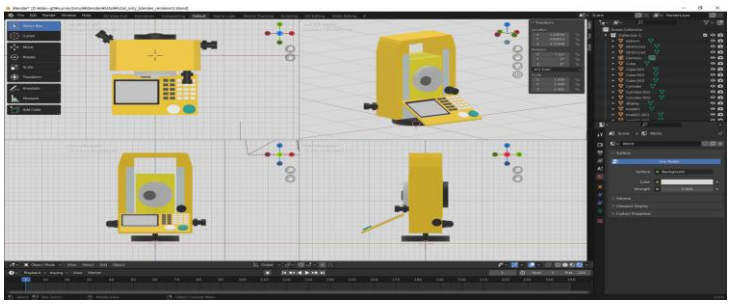
In this paper, “Blender” will be employed as a 3D modeling tool. Blender is a free, open source software, but it provides powerful features, convenience and stability that are

comparable to commercial ones, and also provides a great CYCLE RENDERER that is indistinguishable from real life. In this paper, modelings of total stations, tripods, and pole targets are imported into the Unreal Engine for application.

2.1. Total station

Total station modeling

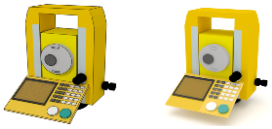
The total station is modeled as follows.



While the lower plate part and the upper part of the total station could be made into a single body, in this paper, the plate part and its mechanical part will be made separately to enable the rotation on the plate.

Total station rendering results

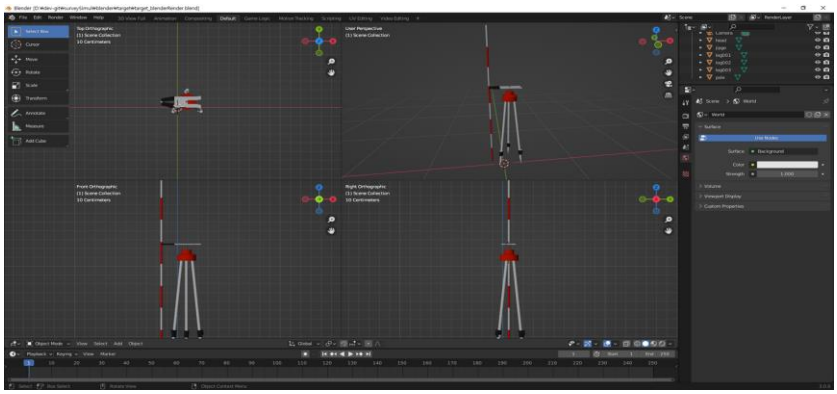
As a result of modeling the total station, the final result was made as follows.



2.2. Pole target

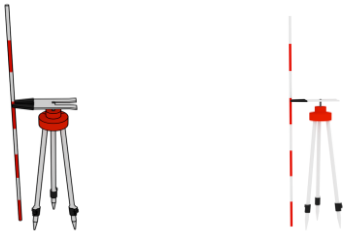
Pole target modeling

The pole target is modeled as follows.



Pole target rendering results

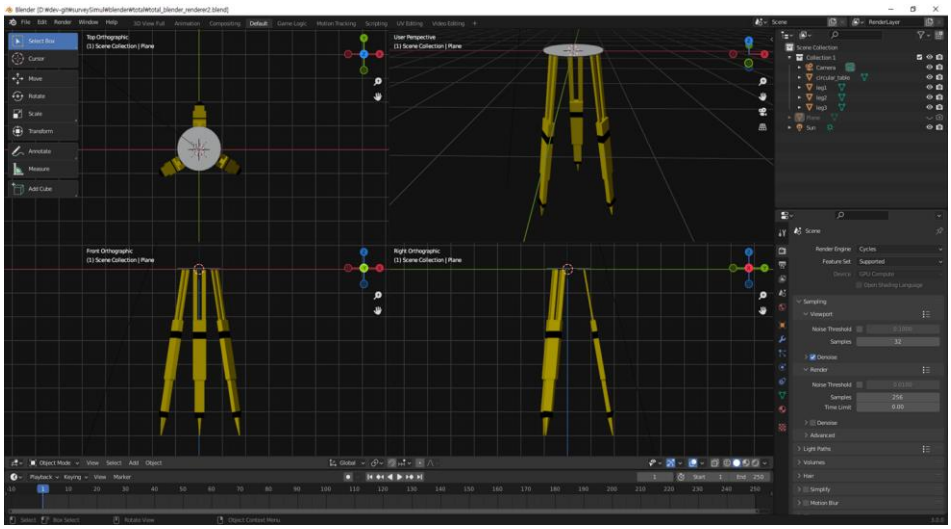
Using the pole target model and texture, the result was created as follows.



2.3. Tripod

Tripod modeling

Tripod is modeled as follows.



Tripod rendering results

Using the tripod model and texture, the result was created as follows.

3. Game engine programming

In this paper, Unity 3D is used as the game engine. Unity 3D is one of the most widely adopted game engines. It is especially used when making lightweight mobile games. Unity 3D is easy to learn by using C# as a base, which is a relatively easy programming language. In particular, rather than developing everything in C# from scratch, it is designed to set a lot of parts as UI and to code only the parts that need scripts in C#, so you can develop quickly and easily.

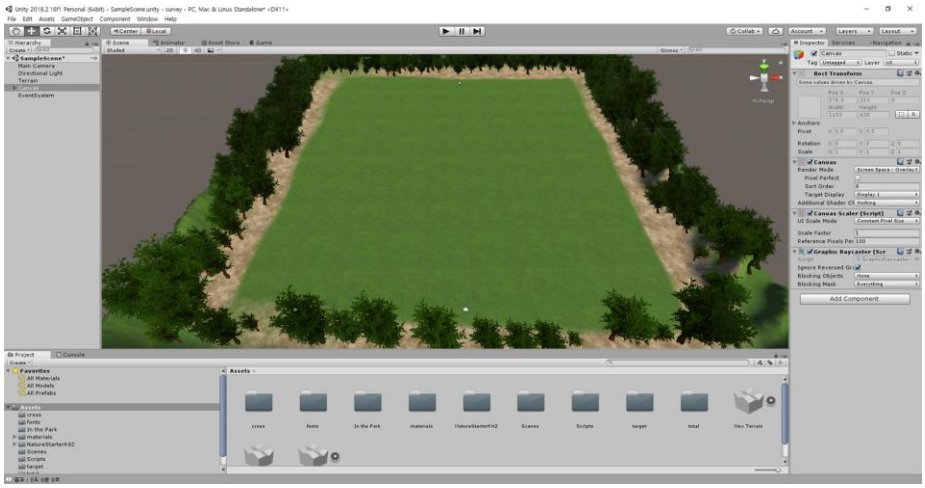
In this paper, the repeating method with Unity 3D will be implemented. The repeating method is done by measuring the same angle three times and averaging.

3.1. Exporting 3D modeled surveying equipment

The first thing to consider when exporting a 3D model is the unit compatibility between the model and the game engine. In fact, you have to consider it from the beginning of modeling, but the combination of Unity 3D and Blender has a great advantage as Unity 3D is able to read native Blender files. Consequently, Blender files can be imported and used immediately without any compatibility concerns.

3.2. Staging a large space for training

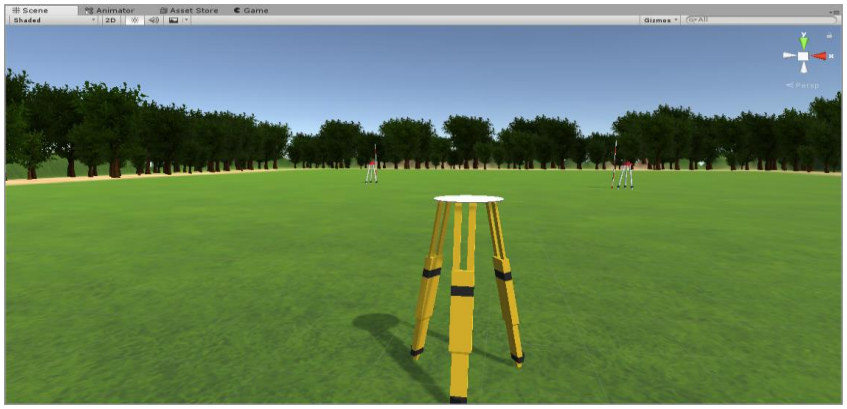
In Unity3D, on top of soil texture, grass is rendered, and trees are planted around it to define the boundary.

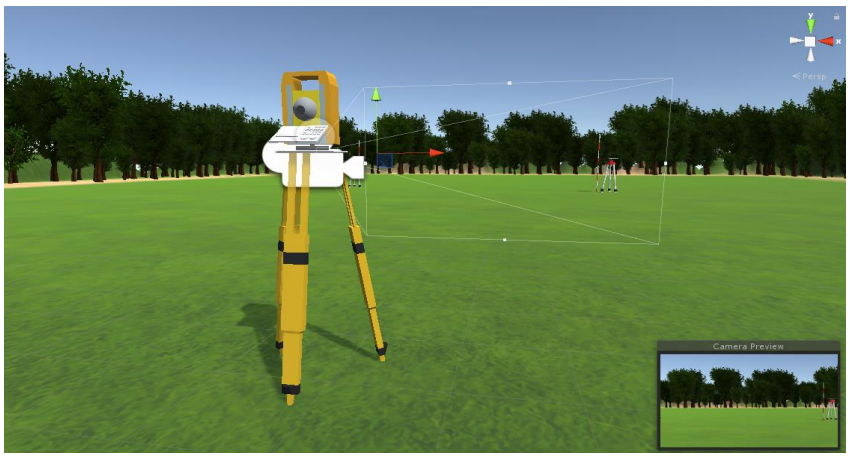


3.3. Importing 3D model of surveying equipment

Unity3D can import models created by Blender directly. When you import the 3D models of the surveying equipment, you can use it immediately in Unity 3D.

The imported assets are arranged as shown in the following figure.





3.4. Coding with C#

Unity3D is designed to add functions by attaching C# code to the game object on the scene. Since this paper will implement the repeating method in this paper, it only requires to add a function to the total station.

In Unity3D, a function Update() is called for each GameObject, and this function is called every frame that the screen is drawn. Unity 3D can use this function to process movements such as relocation and rotation. The rotation of the total station was processed within the Update() function. The following is a part of the Update() function.

```
// Receive rotation value from keyboard or gamepad.
float nHorz1 = Input.GetAxis("Horizontal");
float nVert1 = Input.GetAxis("Vertical");
```

```
// move cross
float nHorz2 = Input.GetAxis("HorizontalTurn");
float nVert2 = Input.GetAxis("VerticalTurn");

Debug.Log("nHorz = " + nHorz + ", nVert = " + nVert);

transform.rotation *= Quaternion.Euler(0, nHorz1 * 1.0f, 0);
transform.rotation *= Quaternion.Euler(0, nHorz2 * 0.005f, 0);
```

In the Update() function, it checks whether the keyboard set for left/right rotation is pressed through the code above. Also, by checking how much the gamepad stick has moved, the rotation value is recognized comprehensively and rotation is processed.

```
// Y button
if (Input.GetButtonDown("Jump"))
{
    Debug.Log("Y Button : Start surveying");

    m_bSurveyMode = !m_bSurveyMode;

    if (true == m_bSurveyMode)
    {
        CamMain.enabled = false;
        CamHead.enabled = true;
    }
    else
    {
        CamHead.enabled = false;
        CamMain.enabled = true;
    }
}
```

Pressing the Y button on the gamepad or the Space bar key on the keyboard start the survey mode viewed through the total station or return to the original state.

The survey mode was implemented by attaching a camera supported by Unity 3D to the total station lens and viewing it with that camera.

```
// Press the A button to measure angle at the current location.
if (Input.GetButtonDown("Fire1"))
{
   if (m_nCurrentPoint < 4)
   {
      m_arVecPoint[m_nCurrentPoint].x = CrossObj.transform.position.x;
      m_arVecPoint[m_nCurrentPoint].y = CrossObj.transform.position.y;
      m_arVecPoint[m_nCurrentPoint].z = CrossObj.transform.position.z;

      Debug.Log("measured: m_nCurrentPoint=" + m_nCurrentPoint + ", x=" + m_arVecPoint[m_nCurrentPoint].x);

      // Calculate the angle.
      float nAngle = this.calcAngle();

      if (1 == m_nCurrentPoint)
      {
         this.m_arUserPoint[0] = nAngle;
         Swing1Text.text = "1 : " + this.getDMS(nAngle);
      }

      if (2 == m_nCurrentPoint)
      {
         this.m_arUserPoint[1] = nAngle;
         Swing2Text.text = "2 : " + this.getDMS(nAngle);
      }

      if (3 == m_nCurrentPoint)
      {
         this.m_arUserPoint[2] = nAngle;
         Swing3Text.text = "3 : " + this.getDMS(nAngle);

         // show on average.
         float nAverage = (this.m_arUserPoint[0] + this.m_arUserPoint[1] + this.m_arUserPoint[2]) / 3.0f;
         UserAnaswerText.text = "Average of 3 measurements :" + this.getDMS(nAverage);
```

```
            // show the answer
            RightAnswerText.text = "actual angle : " + getDMS(this.m_nCorrectAnswer);

            // show the difference
            GapText.text = "difference : " + getDMS(Mathf.Abs(this.m_nCorrectAnswer - nAverage));
        }
        m_nCurrentPoint++;
    }
}
```

By pressing the A button on the gamepad or the Ctrl key on the keyboard, a point is taken for each survey. After taking the second point, the angle is calculated immediately whenever a point is taken and the measured angle is displayed on the screen. This angle is actually a very accurate calculated value.


```
// Calculate the angle.
private float calcAngle()
{
    float nAngle = -1.0f;
    if (1 <= this.m_nCurrentPoint)
    {
        // Distance between previous cross position and camera (side b)
        Vector3 vec1 = new Vector3(m_arVecPoint[m_nCurrentPoint - 1].x, CamHead.transform.position.y,
m_arVecPoint[m_nCurrentPoint - 1].z);
        float nDistance1 = Vector3.Distance(vec1, CamHead.transform.position);
```

```
        // The distance between the current cross position and the camera (side c)
        Vector3 vec2 = new Vector3(m_arVecPoint[m_nCurrentPoint].x, CamHead.transform.position.y,
m_arVecPoint[m_nCurrentPoint].z);
        float nDistance2 = Vector3.Distance(vec2, CamHead.transform.position);

        // The distance between the previous cross position and the current cross position. (a side)
        float nDistance3 = Vector3.Distance(vec1, vec2);
        float nTemp = (nDistance1 * nDistance1 + nDistance2 * nDistance2 - nDistance3 * nDistance3) / (2 * nDistance1 *
nDistance2);
        Debug.Log("nTemp = " + nTemp);

        float nTempA = Mathf.Acos(nTemp);
        Debug.Log("nTempA = " + nTempA);

        // radians are calculated. What we need is a degree.
        float nRadian = Mathf.Acos((nDistance1 * nDistance1 + nDistance2 * nDistance2 - nDistance3 * nDistance3) / (2 *
nDistance1 * nDistance2) );
        Debug.Log("nRadian = " + nRadian);

        nAngle = nRadian * (180.0f / Mathf.PI);
        Debug.Log("nAngle = " + nAngle);
    }
    return nAngle;
}
```

Each time you take a point with the function in the code above, the angle is accurately calculated. As Unity 3D is a game engine using 3D, relatively accurate mathematical calculations are possible.

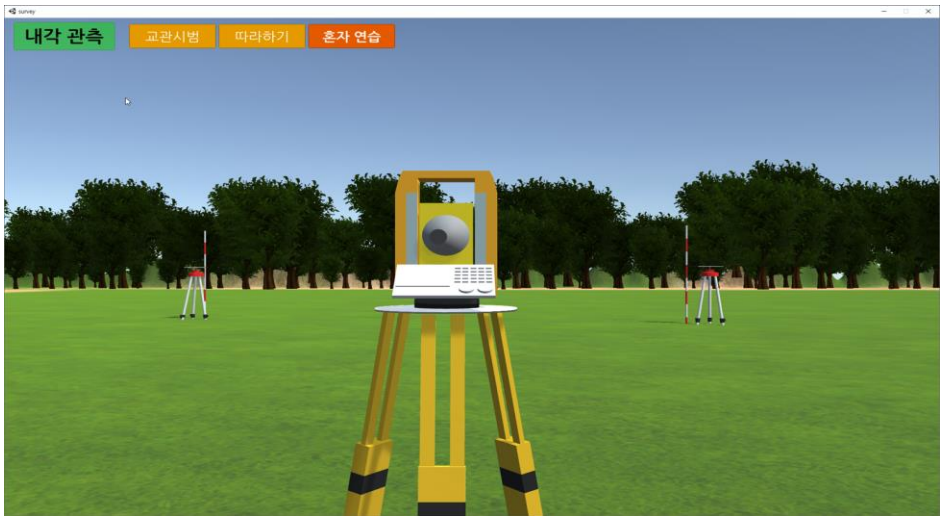
```
// returns the angle in degrees minutes and seconds.
private string getDMS(float nAngle)
{
    // Divide the angle in degrees minutes and seconds.
    float nMinuteTemp = (nAngle - Mathf.Floor(nAngle)) * 60.0f;
    float nSceondTemp = (nMinuteTemp - Mathf.Floor(nMinuteTemp)) * 60.0f;
```

```
    int nDegree = Mathf.FloorToInt(nAngle);
    int nMinute = Mathf.FloorToInt(nMinuteTemp);
    int nSecond = Mathf.FloorToInt(nSceondTemp);

    Debug.Log("result : " + nDegree + "degrees " + nMinute + "minutes " + nSecond + "seconds");

    return (nDegree + "degrees " + nMinute + "minutes " + nSecond + "seconds");
}
```

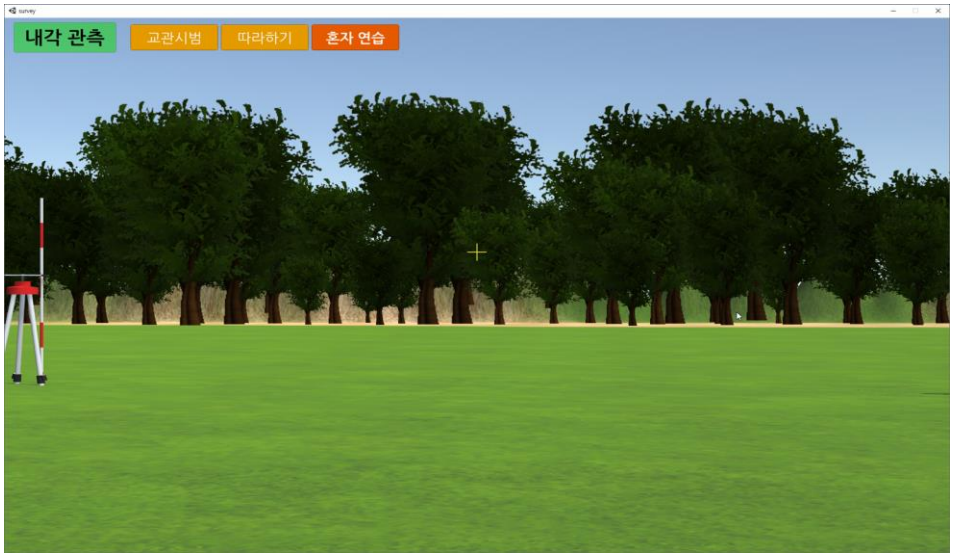
The entire code is so simple as above. Only three functions were used: the Update() function, the calcAngle() function, and the getDMS() function. Since Unity 3D performs heavy processing with 3D graphics, developers only need to describe the remaining features.

4. Usage

Navigating the program made with the Unity game engine is as follows.

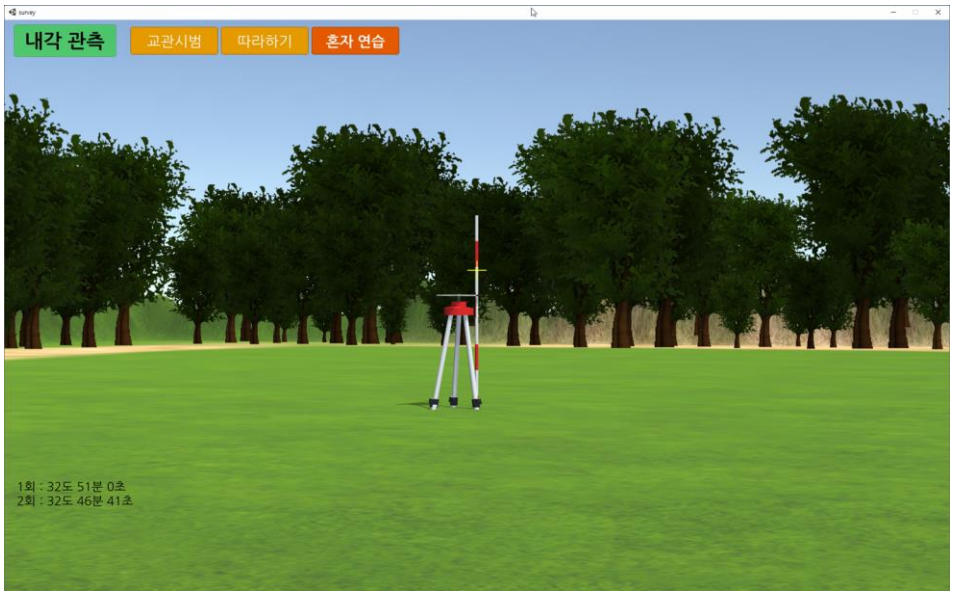


1. Launch the program.



2. Press the Y button on the gamepad or press the space bar on keyboard to enter the survey mode (viewed through the total station).

3. After moving to the left pole using a gamepad or keyboard, press the A button on gamepad or ctrl key on keyboard to take the first point. The angle between the first pole (Figure 2) and the second pole (Figure 3) is calculated and shown in the lower left.



4. After moving to the left pole using a gamepad or keyboard, press the A button on gamepad or ctrl key on the keyboard to take the second point. Then, the angle between the picture 3 above and the second point taken this time is shown in the lower left corner.

5. Finally, when the third point is taken, the angle between the 4 above and the point taken

this time is calculated and displayed, and the difference is shown by how much difference it is from the actual angle.

In this way, we can explain easily to a survey engineer trainee how to measure an angle with a repeating method without any real equipment or place.

5. Conclusion

For survey training, it generally requires significant resources including a large area and costly surveying equipment. In this paper, to solve this problem, a virtual training system using the game engine was presented. An innovative education system brought by game engines like Unity 3D is going to make a difference. In addition, by making each training course a separate module, you can teach a variety of different situations. By using this method to train survey engineers, it is expected that the cost, time, and convenience challenges can be addressed.

BIOGRAPHICAL NOTES

CONTACTS

Mr. Moonsik Kim
LX Education Institute
182 Yeonsudanji-gil, Sagok-myeon, Gongju-si, Chungcheongnam-do, South Korea (32522)
Gongju
South Korea
Tel. +82 41 401 0205
Email mskim@lx.or.kr
Web site: www.lx.or.kr