

3D콘텐츠 이론 및 활용

유길상

4주. C# 스크립트

1,2교시 : 변수, 조건문

3,4교시 : 반복문, 변수범위, 배열, 키입력

학습 목표

- 변수의 의미를 이해하고 활용할 수 있다.
- 연산자의 의미를 이해하고 활용할 수 있다.
- 조건문을 이해하고 활용할 수 있다.

학습 내용

- 변수
- 연산자
- IF 문

1. 변수

1) 변수란?

- 어떤 숫자나 문자열을 저장할 수 있는 상자
 - 자료형 : 변수의 형태
 - ex) int - 정수형, float - 실수형, string - 문자열, bool- 참/거짓 판정



1. 변수

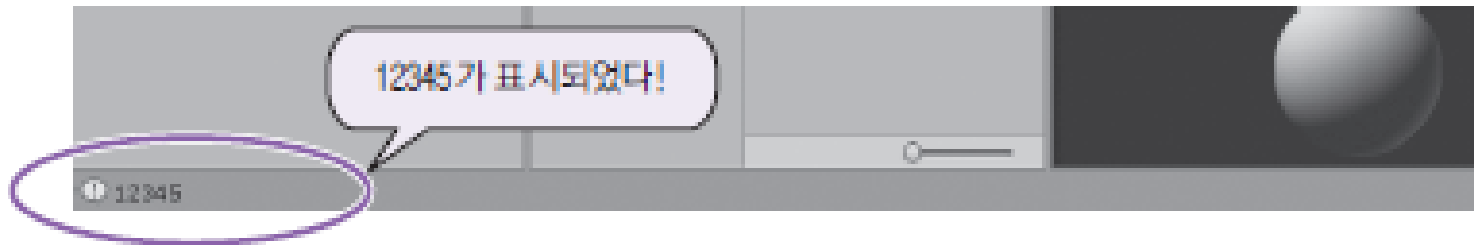
2) 변수 사용하기

■ 정수형

- 자료형 : 변수의 형태
- ex) int - 정수형, float - 실수형, string - 문자열, bool- 참/거짓 판정

```
void Start () {
    int a = 12345;

    print ("Welcome to Unity World!!!");
    Debug.Log (" 저장된 값은 " + a);
}
```



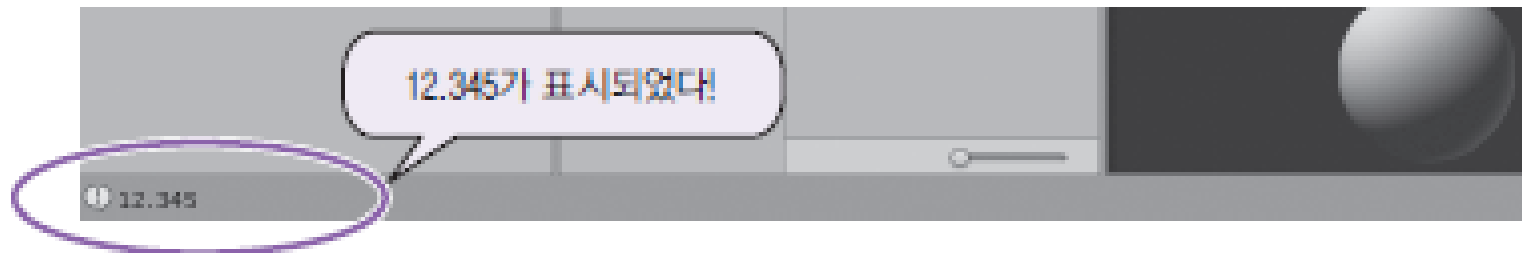
1. 변수

2) 변수 사용하기

- 실수형

```
void Start () {
    float a = 12.345f;

    print ("Welcome to Unity World!!!");
    Debug.Log (" 저장된 값은 " + a);
}
```



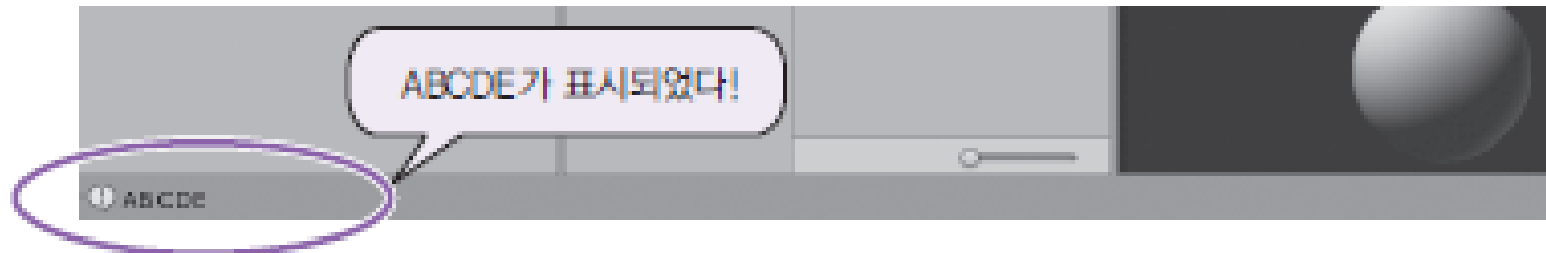
1. 변수

2) 변수 사용하기

■ 문자형

```
void Start () {
    string a = "ABCDE";

    print ("Welcome to Unity World!!!");
    Debug.Log (" 저장된 값은 " + a);
}
```



1. 변수

3) 변수의 특성

- 변수의 내용은 원하는 대로 바꿀 수 있다.

```

void Start () {
    int a = 1;
    Debug.Log (a);
    a = 2;
    Debug.Log (a);
    a = 3;
    Debug.Log (a);
}
    
```

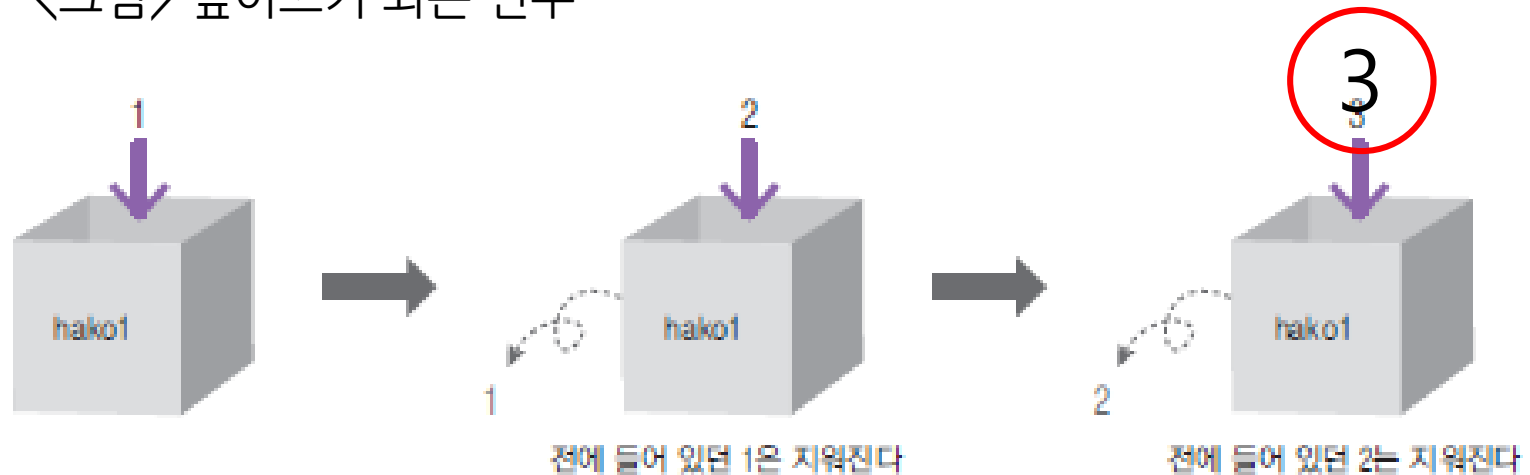
- 위 코드를 실행했을 때 변수 **a**에 저장된 값은?

1. 변수

3) 변수의 특성

- 변수의 내용은 원하는 대로 바꿀 수 있다. => **대입**

〈그림〉 덮어쓰기 되는 변수

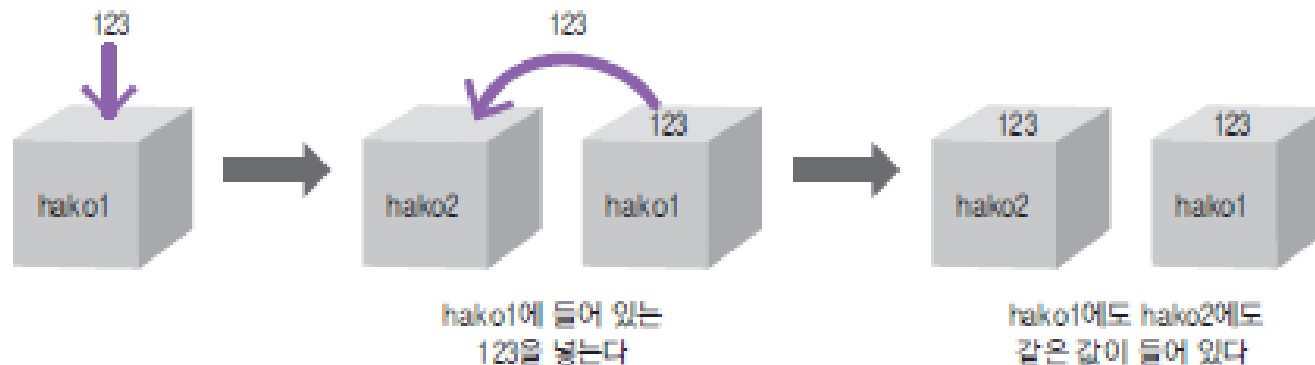


1. 변수

3) 변수의 특성

- 변수에 변수를 대입할 수 있다

```
void Start () {
    int a;
    int b;
    a = 123;
    b = a;
    Debug.Log (b);
}
```



2. 변수의 연산

1) 사칙연산

- +, -, *, / 를 이용해 산술연산을 할 수 있다.
- 곱셈과 나눗셈 연산을 할 때는 자료형을 주의해야 한다.

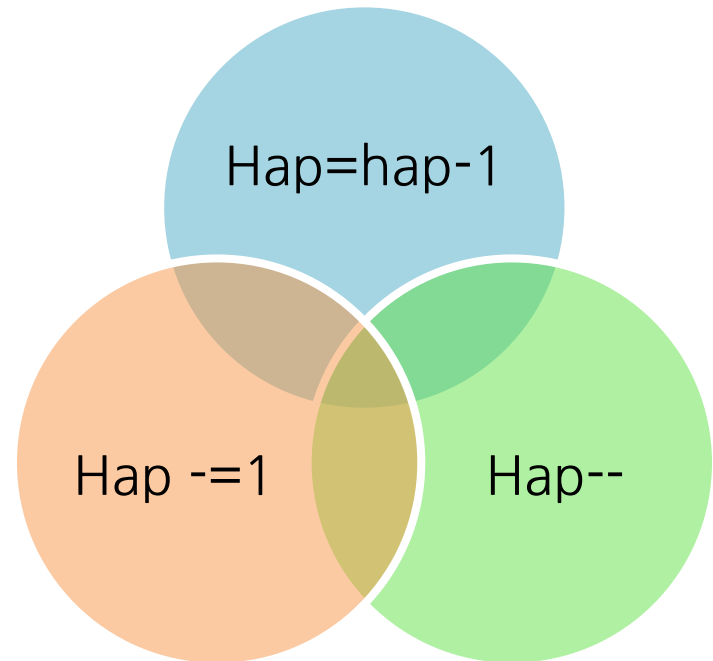
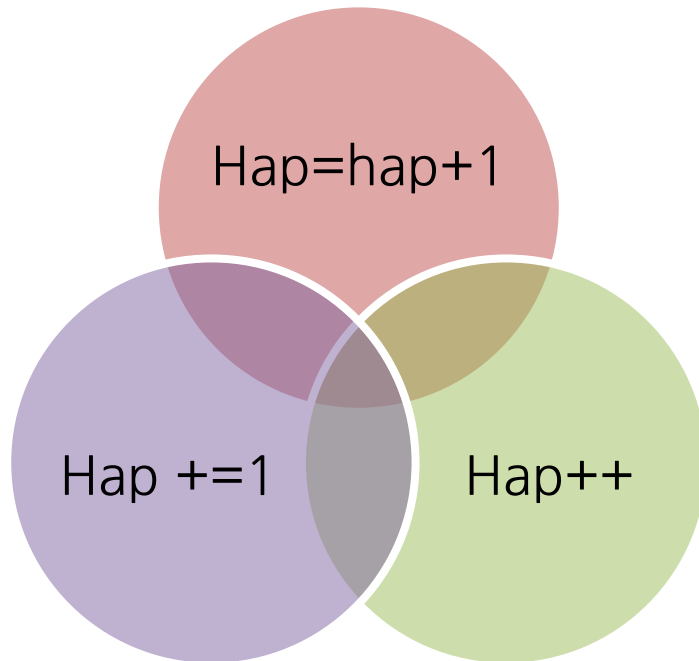
```

void Start () {
    int a;
    int b;
    a = 2 * 3; // 곱하기
    b = 10 / 2; // 나누기
    Debug.Log (a);
    Debug.Log (b);
}
    
```

2. 변수의 연산

2) 증감연산

- ++, --로 증가, 감소연산을 쉽게 코딩 할 수 있다.
- `Hap = 1; // Hap 변수에 저장된 값이 1일 경우 1씩 증가, 감소연산 예`



2. 변수의 연산

3) 누적연산

- 복합대입연산자를 통해 짧은 코딩으로 작성할 수 있다.
- Hap = Hap + 10;
- Hap = Hap - 1;
- Hap = Hap * 4;
- Hap = Hap / 6;

```
Hap = 1;
Hap += 10;
```

11

```
Hap = 5;
Hap -= 1;
```

4

```
Hap = 10;
Hap *= 4;
```

40

```
Hap = 24;
Hap /= 6;
```

4

3. 조건문

if 문: 지정한 조건에 맞으면 프로그램 구문을 실행한다.

```

void Start () {
    int a;
    a = 1;

    if (a == 1) {
        Debug.Log ("조건에 맞습니다.");
    }
}

```

// 조건에 부합하면 코드 블록을 빠져 나온다.

Hap == 1	Hap의 값이 1 이면
Hap >= 1	Hap의 값이 1 이상이면
Hap <= 1	Hap의 값이 1 이하이면
Hap != 1	Hap의 값이 1 이 아니면

3. 조건문

- if ~ else : 조건이 맞지 않을 때, 실행할 수 있는 구문을 추가할 수 있다.

studyScript.cs

```
void Start() {
    int hako = 1;

    if(hako == 1) { // hako의 값이 1이면.
        Debug.Log("1이야");
    } else { // 그렇지 않으면.
        Debug.Log("1이 아니야");
    }
}
```

studyScript.cs

```
void Start() {
    int hako = 1;

    if(hako == 1) { // hako의 값이 1이면.
        Debug.Log("1이야");
    } else if(hako == 2) { // 그렇지 않고 2라면.
        Debug.Log("1이 아니라 2야");
    } else { // 모두 해당되지 않으면.
        Debug.Log("1도 2도 아니라 3이야");
    }
}
```

else에 if를 더해서 여러 조건에 적용시킬 수 있다.

3. 조건문(switch ~ case)

```
void Start () {
    int a = 3;

    switch(a){
    case 1 :
        Debug.Log ("1이군요");
        break;
    case 2 :
        Debug.Log ("2이군요");
        break;
    case 3 :
        Debug.Log ("3이군요");
        break;
    default:
        Debug.Log ("모릅니다");
        break;
    }
}
```

Keypoint

1. case 다음에 공백 넣고 값
2. 값 뒤에 콜론(:)으로 구분
case 1 :
3. case 구문 마지막에는 break
4. 데이터형에 따라 ‘ ‘ 필요
ex, char a='a';
case 'a':

실습.

```

void Start () {
    string name;
    name = "Korea";
    Debug.Log (name);
}
    
```

```

void Start () {
    int answer = 10;
    answer += 5;
    Debug.Log (answer);
}
    
```


3D콘텐츠 이론 및 활용

유길상

4주. C# 스크립트

1,2교시 : 변수, 조건문

3,4교시 : 반복문, 변수범위, 배열, 키입력

3, 4교시 학습

- 반복문에 대하여 이해하고 활용할 수 있다.
- 변수가 참조할 수 있는 유효범위에 대하여 이해하고 활용할 수 있다.
- 배열에 대하여 이해하고 적용할 수 있다.
- 벡터형 변수에 대하여 이해하고 활용할 수 있다.
- 키보드 입력에 따라 객체를 이동시킬 수 있다.

3, 4교시 학습내용

- 반복문
- 변수의 유효 범위
- 배열
- Vector형 변수
- 키입력처리

1. 반복문

- 출력결과는?

```
void Start () {
    for (int i = 0; i < 10; i+=2) {
        Debug.Log (i);
    }
} // 0, 2, 4, 6, 8
```

- 10 에서 시작하여 1씩 감소하면서 1까지 출력하기

```
void Start () {
    for (int i = 10; i > 0; i--) {
        Debug.Log (i);
    }
}
```

1. 반복문

1부터 10000까지의 합은? 50005000

```

void Start ()
{
    int sum = 0;
    for (int i = 1; i <= 10000; i++) {
        sum = sum + i; // sum += i;
    }

    Debug.Log ("1부터 10000까지의 합은? " + sum);
}
    
```

sum = 0;

누적되는 값으로 사용되는 변수는 초기화를 하지 않을 경우 잘못된 결과값이 나올 수 있다.

2. 변수의 유효범위

- 프로그램내에서 유효범위 : 변수가 참조할 수 있는 범위

```

void Start ()
{
    int sum = 0;
    for (int i = 1; i <= 10000; i++) {
        sum = sum + i; // sum += i;
    }
    Debug.Log (i);
    Debug.Log ("1부터 10000까지의 합은? " + sum);
}
    
```

* 변수의 유효범위는 코드블록 내에서만 활용된다.

2. 변수의 유효범위

- 프로그램내에서 유효범위 : 변수가 참조할 수 있는 범위

```

void Start () {
    int x = 1;
    if (x == 1) {
        int y = 2;
        Debug.Log (x);
        Debug.Log (y);
    }
    Debug.Log (y);
}
    
```

2. 변수의 유효범위

- 변수 i의 유효범위는 클래스 내에서 모두 사용할 수 있다.

```
public class basic : MonoBehaviour
{
    int i = 0;

    void Start ()
    {
        int sum = 0;
        for (i = 1; i <= 10000; i++) {
            sum = sum + i; // sum += i;
        }
        i--;
        Debug.Log ("1부터 " + i + "까지의 합은? " + sum);
    }

    void Update ()
    {
    }
}
```


2. 변수의 유효범위

- 유효범위(외부/내부) : 변수가 참조할 수 있는 범위
 - public : 프로그램 외부에서 내용을 변경할 수 있는 변수
 - private : 외부에서 보이지 않고 내부에서만 변경할 수도 있는 변수

```
public class basic : MonoBehaviour
```

```
{
```

```
    int i = 0;
```

```
    int count = 10000;
```

```
    void Start ()
```

```
{
```

```
        int sum = 0;
```

```
        for (i = 1; i <= count; i++) {
            sum += i;
        }
```

```
        Debug.Log ("1부터 " + count + "까지의 합은? " + sum);
```

```
    }
```

```
}
```



2. 변수의 유효범위

```

public string BloodType;
// Use this for initialization
void Start () {
    switch(BloodType){
        case "A":
            Debug.Log ("A형 이군요...");
            break;
        case "B" :
            Debug.Log ("B이 이군요...");
            break;
        case "O" :
            Debug.Log ("O이 이군요...");
            break;
        default:
            Debug.Log ("A B O형이 아니군요");
            break;
    }
}
    
```

3. 배열

- 비슷한 변수의 사용이 많을 경우

```

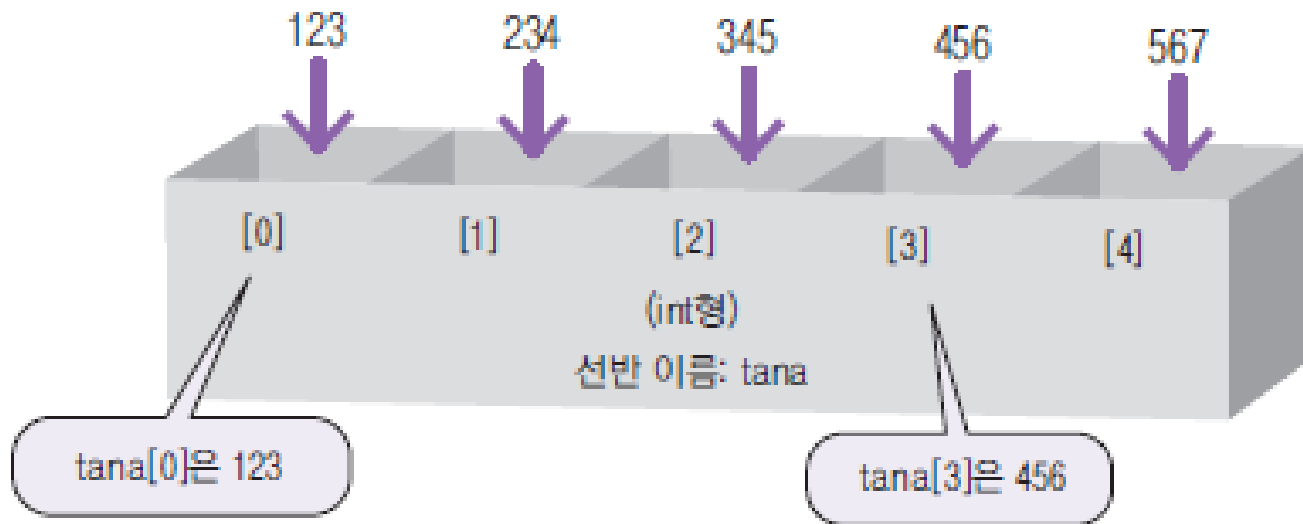
int Object_00;
int Object_01;
int Object_02;
int Object_03;
int Object_04;
int Object_05;
int Object_06;
int Object_07;
.
.
int Object_95;
int Object_96;
int Object_97;
int Object_98;
int Object_99;
    
```

3. 배열

1) 배열이란

- 여러 개의 변수를 하나의 변수로 사용하여 데이터를 사용(관리)할 수 있다.
- 배열의 방번호는 **0으로 시작**

```
int[] tana = new int[5];
```



3. 배열

2) 배열선언

- 방법 1. 선언과 동시에 값을 넣어준다.
- 방법 2. 빈 배열을 선언 후 나중에 값을 넣어준다.

```

public class basic : MonoBehaviour
{
    int[] array1 = new int[10];
    int[] array2 = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

    void Start ()
    {
        array1 [1] = 1;
        Debug.Log (array1 [1]);    // 1
        Debug.Log (array2 [1]);    // 2
    }
}
    
```

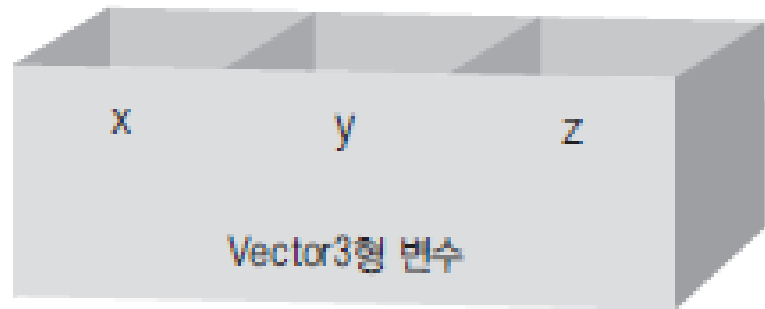
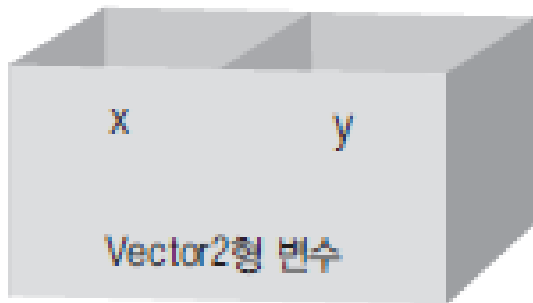
3. 배열

반복문을 이용한 배열 값 확인

```
void Start ()  
{  
    int[] array2 = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };  
    for (int i = 0; i < 10; i++) {  
        Debug.Log (array2 [i]);  
    }  
}
```

4. Vector형 변수

- vector2, vector3 - 2D, 3D를 표현할 수 있는 자료형 중 하나
- 캐릭터가 3D 공간(또는 2D) 어디에 있고, 어디에 진행하려 하는지,
- 어느 방향으로 힘을 가하려 하는지 등 캐릭터의 움직임을 나타내는 데 유용.



<벡터2형 변수와 벡터3형 변수 비교>

4. Vector형 변수

```

void Start ()
{
    Vector3 location;
    location = gameObject.transform.position;
    Debug.Log (location);
    Debug.Log (location.x);
    Debug.Log (location.y);
    Debug.Log (location.z);
}
    
```


5. 키보드와 마우스 입력받기

1) 키보드 입력

- “Input.동작(키코드)”의 형태로 사용한다.

Input.GetKey : 키가 눌린 상태

Input.GetKeyDown : 키가 눌러진 순간에 한번 작동

Input.GetKeyUp : 키를 올라가는 순간에 한번 작동

Input.AnyKey : 아무 키라도 눌러진 상태에서는 계속 작동

Input.AnyKeyDown : 아무 키가 눌리는 순간에 한번 작동



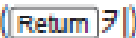






- 활용 예

```
void Update ()
{
    if (Input.GetKeyUp (KeyCode.LeftArrow)) {
        transform.Translate (-1, 0, 0);
    } else if (Input.GetKeyUp (KeyCode.RightArrow)) {
        transform.Translate (1, 0, 0);
    }
}
```

5. 키보드와 마우스 입력받기

1) 키보드 입력

■ Key code

키코드	키
KeyCode.Space	
KeyCode.Return	 키 ( 키)
KeyCode.UpArrow	 키
KeyCode.DownArrow	 키
KeyCode.LeftArrow	 키
KeyCode.RightArrow	 키
KeyCode.Escape	 키
KeyCode.Backspace	 키

키코드	키
KeyCode.X	 키
KeyCode.S	 키
KeyCode.LeftShift	왼쪽  키
KeyCode.RightShift	오른쪽  키
KeyCode.LeftControl	왼쪽  키
KeyCode.RightControl	오른쪽  키
KeyCode.Alpha1	 키
KeyCode.F1	 키

5. 키보드와 마우스 입력받기

2) 마우스 입력

- (0) = 좌클릭, (1) = 우클릭

Input.GetMouseButton(0) : 마우스 좌클릭 되었을 때 계속체크

Input.GetMouseButtonDown(0) : (좌)클릭 되었을 때 한번 만 체크

Input.GetMouseButtonUp(0)

- 활용 예

- Input.mousePosition : 마우스 포인터의 위치를 구한다.

```

void Update ()
{
    if (Input.GetMouseButtonDown(0)) {
        Debug.Log (Input.mousePosition);
    }
}
    
```

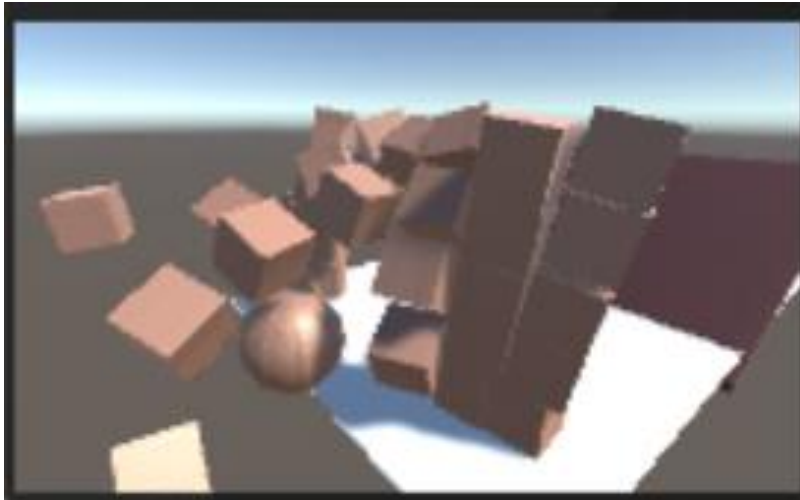
과제(배점 20점)

1

중력이 적용된 공을 이용한 박스 넘어뜨리기
- 플레이 버튼을 클릭 후 스페이스 바를 누르면 공이 떨어지도록 처리

2

포탑 만들기



참고이미지

<http://shop.youngjin.com/goods/view?no=100006656>

포탑

30분 1 2 3 4 완전 초보

여러분의 성은 크리퍼와 좀비 그리고 다른 침략자들을 막기 위하여 그들이 접근하는 것을 볼 수 있어야 합니다.

여러분의 요새에 우선 배치할 것은 석재 벽돌 블록으로 만든 포탑입니다.

포탑은 넓은 영역을 관찰할 수 있도록 도와주며, 먼 거리의 적을 식별할 수 있습니다.

요새를 둘러싼 벽들이 교차하는 모퉁이에 포탑을 추가하여 왕국 전체를 살펴 볼 수 있습니다.

포탑은 탑 형식으로 성벽의 위쪽으로 넓게 확장하거나 커다란 탑 꼭대기에 짓습니다.

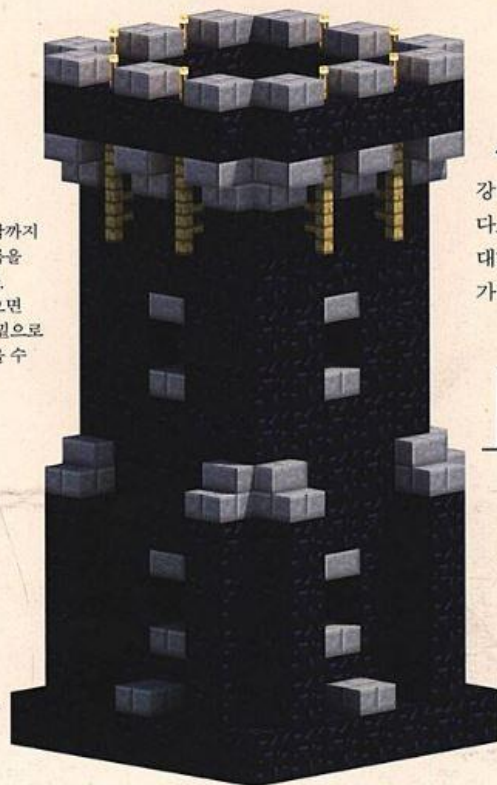
포탑과 성의 다른 구조물에는 궁수들이 안전하게 보호를 받으며 활을 쏠 수 있도록 작고 좁은 구멍이 연속적으로 배치되어 있습니다.



참고이미지

지금까지 여러분은 기본적인 포탑을 만들었습니다. 이제 이것을 좀더 창의적으로 변경해 보겠습니다.
포탑의 모양을 바꾸기 위해, 사용한 블록을 변경하고 다양한 기능을 추가해서 독창적인 모습으로 만들 수 있습니다.
심지어 여러분의 성벽 모퉁이의 모든 포탑을 서로 다른 스타일을 가지도록 할 수도 있습니다.

포탑의 바닥까지
흑요석 블록을
사용합니다.
그렇지 않으면
적들이 그 밑으로
터널을 뚫을 수
있습니다.



i 초강력 벽

강력한 포탑을 만들기 위해서 벽돌 반 블록을 흑요석으로 대신 합니다. 흑요석은 마인크래프트 세계에서 가장 강력한 블록이며, 폭발에 대한 저항이 가장 높습니다. 따라서, 크리퍼의 폭발이나 TNT 공격도 가볍게 견뎌냅니다.

흑요석은 반 블록이나
계단 블록이 없습니다.
따라서, 벽돌 반 블록들을
사용해야 합니다.

i 포탑 지붕

포탑에 지붕을 추가합니다. 높이나 모양은 상관없지만, 적들이 경비중인 병사들을 공격하지 못하거나 공격할 기회가 최소화되도록 만들어야 합니다. 지붕은 병사들이 살아남아 성을 방어하기 쉽도록 합니다.

접근하는 적들을 공격하기 위한
발사대를 만듭니다.



포탑 꼭대기의 가장
자리에 조약돌 벽 블
록을 추가하여 병사
들이 아래로 떨어지
지 않도록 합니다.

참고이미지

