Transform
(Chapter 4-1)

Jin-Mo Kim

jinmo.kim@hansung.ac.kr

Transform

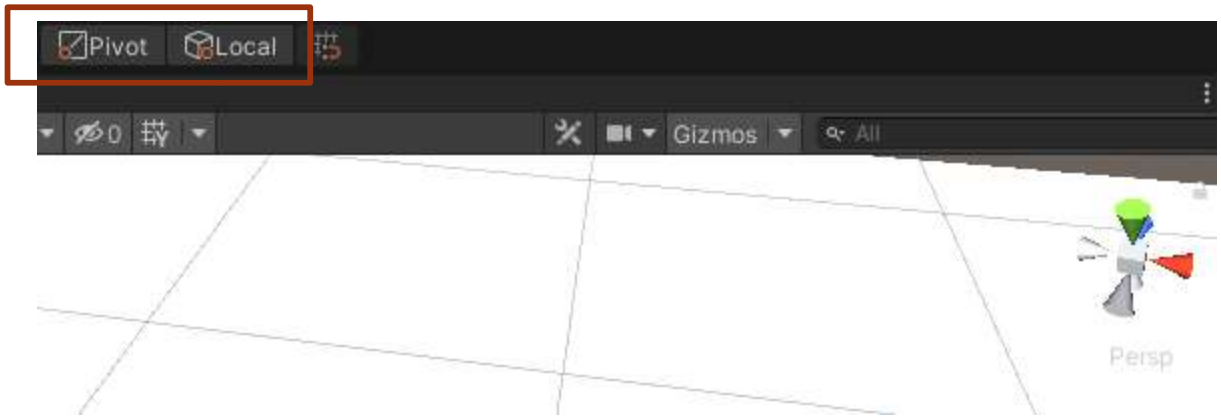
- Transform의 개념 및 활용
  - position: 게임 오브젝트의 위치
  - rotation: 회전을 통한 게임 오브젝트의 방향
  - scale: 게임 오브젝트의 크기
  - Translate: 게임 오브젝트 이동 함수
  - Rotate: 게임 오브젝트 회전 함수

  - transform.forward: 현재 게임 오브젝트의 전방향 벡터(z축)
  - transform.right(좌/우 x축), transform.up(상/하 y축)
    - Vector3.forward: 월드 공간을 기준으로 z축

  - 3차원 속성으로 (x, y, z)의 값을 가지며, 유니티 엔진에서는 Vector3 클래스 변수를 통해 처리
    - 2차원 콘텐츠 제작 시에는 z값은 사용하지 않음

Transform

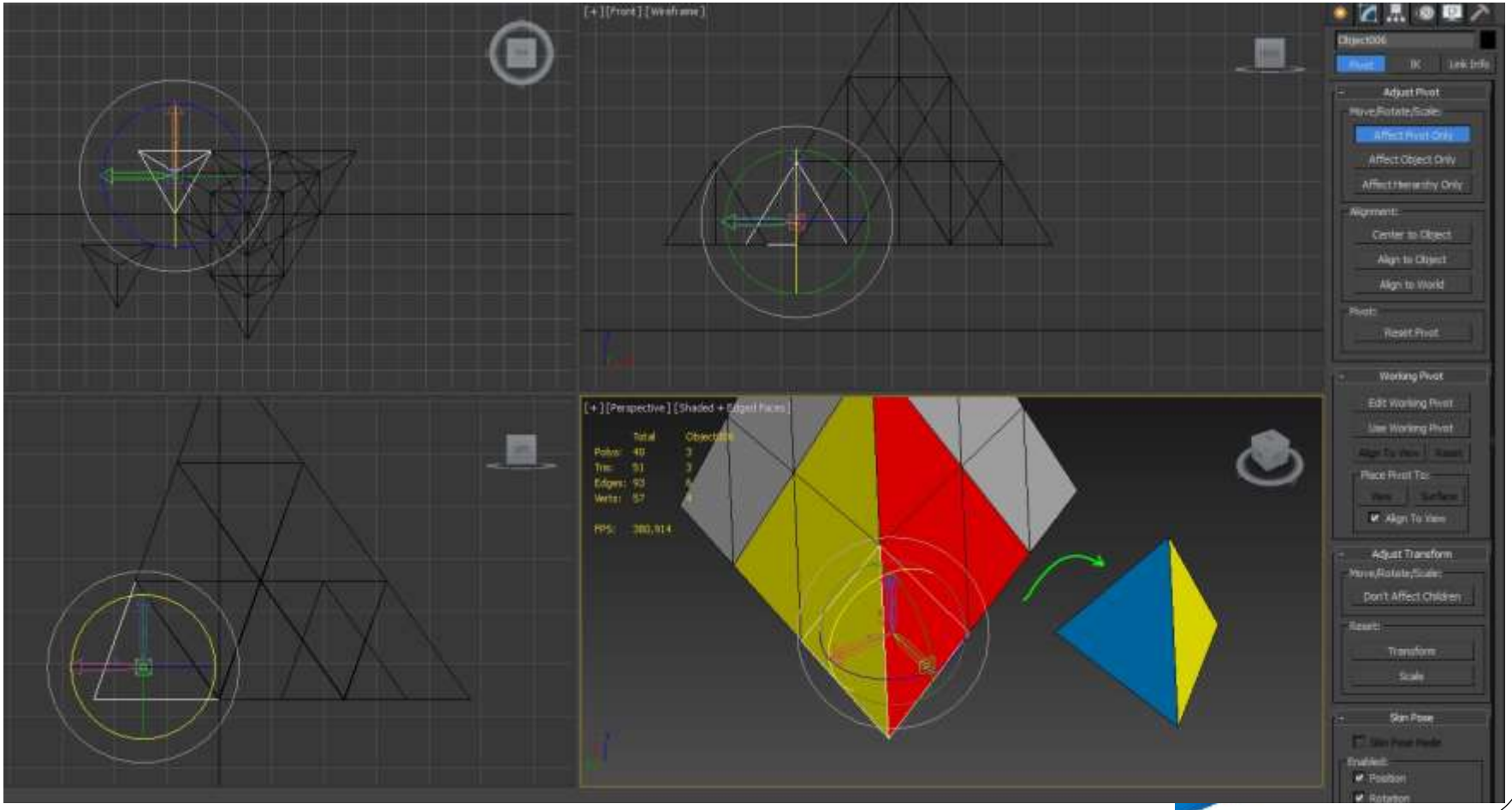
- Transform의 개념 및 활용
  - 유니티 엔진



  - Pivot/Center : 게임 오브젝트의 고유 축 또는 중심을 기준
  - Local/Global : 게임 오브젝트 또는 월드 공간을 기준

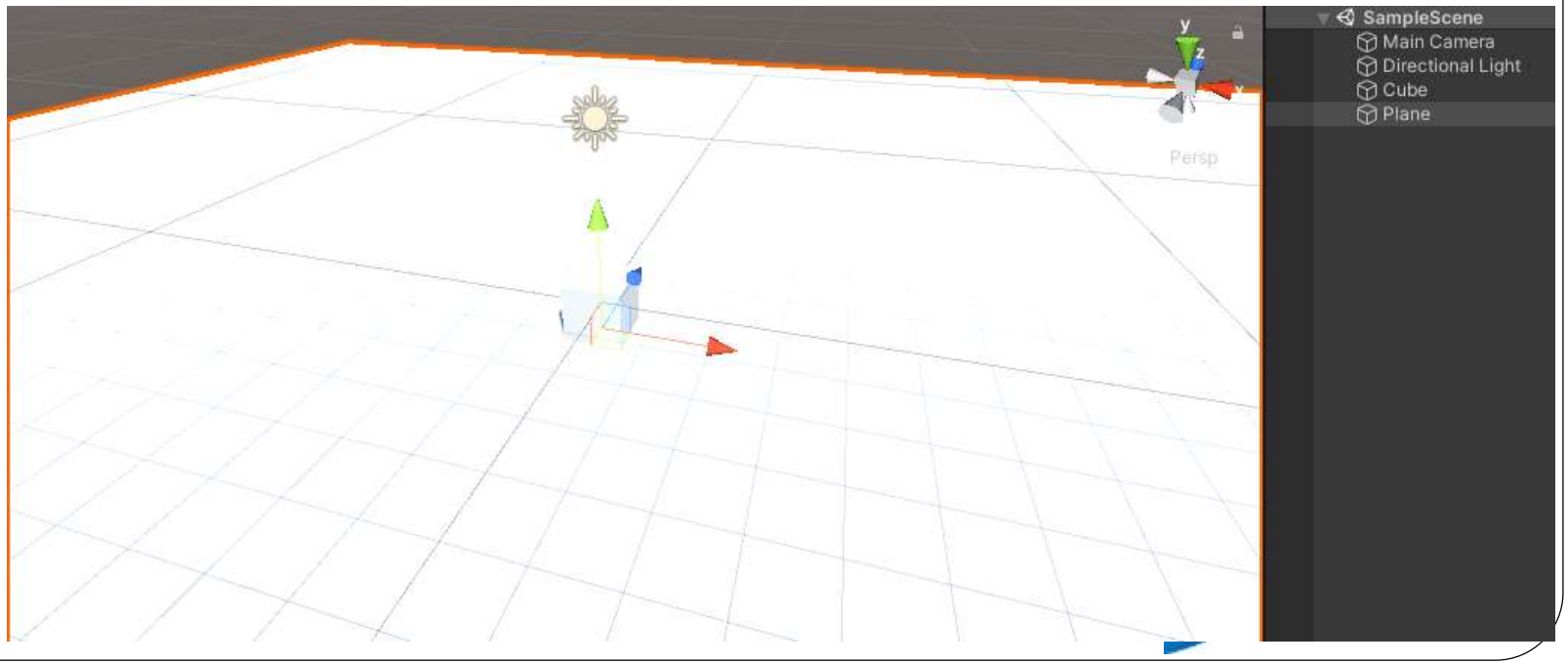
Transform

- Transform의 개념 및 활용
  - Pivot



Transform

- Translate 함수의 활용
  - 간단한 장면 생성
    - Cube
    - Plane
      - position: 0, -0.5, 0
      - scale: 5, 1, 5



Transform

- Translate 함수의 활용
  - Cube 제어 스크립트
    - cshTransform.cs

```
void Update()
    {
        if (Input.GetKey(KeyCode.UpArrow))
        {
            transform.Translate(transform.forward * 2.0f * Time.deltaTime);
        }
        if (Input.GetKey(KeyCode.DownArrow))
        {
            transform.Translate(transform.forward * -2.0f * Time.deltaTime);
        }
        if (Input.GetKey(KeyCode.RightArrow))
        {
            transform.Translate(transform.right * 2.0f * Time.deltaTime);
        }
        if (Input.GetKey(KeyCode.LeftArrow))
        {
            transform.Translate(transform.right * -2.0f * Time.deltaTime);
        }
    }
```

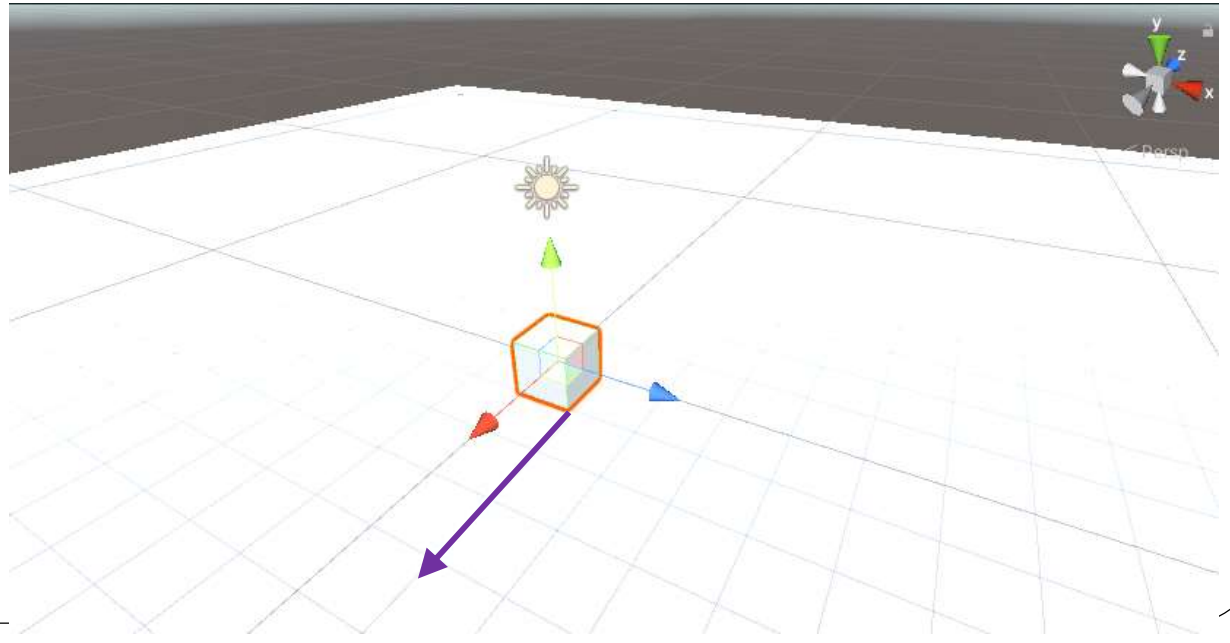
Transform

- Translate 함수의 이해
  - 이동 방향을 바라보도록 설정: LookAt 함수 ***잘못된 사용**

```
void Update()
    {
        Vector3 dir = Vector3.zero;
        if (Input.GetKey(KeyCode.UpArrow))
        {
            transform.Translate(transform.forward * 2.0f * Time.deltaTime);
            dir += transform.forward;
        }
        if (Input.GetKey(KeyCode.DownArrow))
        {
            transform.Translate(transform.forward * -2.0f * Time.deltaTime);
            dir += -transform.forward;
        }
        if (Input.GetKey(KeyCode.RightArrow))
        {
            transform.Translate(transform.right * 2.0f * Time.deltaTime);
            dir += transform.right;
        }
        if (Input.GetKey(KeyCode.LeftArrow))
        {
            transform.Translate(transform.right * -2.0f * Time.deltaTime);
            dir += -transform.right;
        }
        dir = dir.normalized;
        if (dir.magnitude > 0.5f)
        {
            transform.LookAt(transform.position + dir);
        }
    }
```

Transform

- Translate 함수의 이해
  - Translate 함수는 현재 게임 오브젝트를 기준으로 이동하는 함수
  - transform.forward
    - 월드 공간을 기준으로 한 현재 게임 오브젝트의 앞 방향 벡터
    - 아래의 경우, transform.forward는 (1,0,0)
    - 이때, Translate(transform.forward)를 하면,
      - 게임 오브젝트의 x축 방향(1,0,0)으로 이동하게 됨



Transform

- Translate 함수의 이해
  - 이동 방향으로 바라보는 LookAt 함수는 월드 공간을 기준으로
  - Translate를 통한 이동은 바라 보는 방향(forward)으로 이동

```
void Update()
    {
        Vector3 dir = Vector3.zero;
        if (Input.GetKey(KeyCode.UpArrow))
        {
            transform.Translate(Vector3.forward * 2.0f * Time.deltaTime);
            dir += Vector3.forward;
        }
        if (Input.GetKey(KeyCode.DownArrow))
        {
            transform.Translate(Vector3.forward * 2.0f * Time.deltaTime);
            dir += Vector3.back;
        }
        if (Input.GetKey(KeyCode.RightArrow))
        {
            transform.Translate(Vector3.forward * 2.0f * Time.deltaTime);
            dir += Vector3.right;
        }
        if (Input.GetKey(KeyCode.LeftArrow))
        {
            transform.Translate(Vector3.forward * 2.0f * Time.deltaTime);
            dir += Vector3.left;
        }
        dir = dir.normalized;
        if (dir.magnitude > 0.5f)
        {
            transform.LookAt(transform.position + dir);
        }
    }
```

Transform

- Translate 함수의 이해
  - Translate 함수를 월드 공간 중심으로 해석

```
void Update()
    {
        Vector3 dir = Vector3.zero;
        if (Input.GetKey(KeyCode.UpArrow))
        {
            transform.Translate(Vector3.forward * 2.0f * Time.deltaTime, Space.World);
            dir += Vector3.forward;
        }
        if (Input.GetKey(KeyCode.DownArrow))
        {
            transform.Translate(Vector3.back * 2.0f * Time.deltaTime, Space.World);
            dir += Vector3.back;
        }
        if (Input.GetKey(KeyCode.RightArrow))
        {
            transform.Translate(Vector3.right * 2.0f * Time.deltaTime, Space.World);
            dir += Vector3.right;
        }
        if (Input.GetKey(KeyCode.LeftArrow))
        {
            transform.Translate(Vector3.left * 2.0f * Time.deltaTime, Space.World);
            dir += Vector3.left;
        }
        dir = dir.normalized;
        if (dir.magnitude > 0.5f)
        {
            transform.LookAt(transform.position + dir);
        }
    }
```