

셰이더 프로그래밍

Lecture 5

이택희

지난 시간

- ◆ 프래그먼트 셰이더
- ◆ Storage Qualifier
- ◆ 버텍스 셰이더 입력 데이터 패킹

개요

- ◇ 버텍스 셰이더 사용 애니메이션
- ◇ 프래그먼트 셰이더 사용 애니메이션
- ◇ 실습

버텍스 셰이더 사용 애니메이션

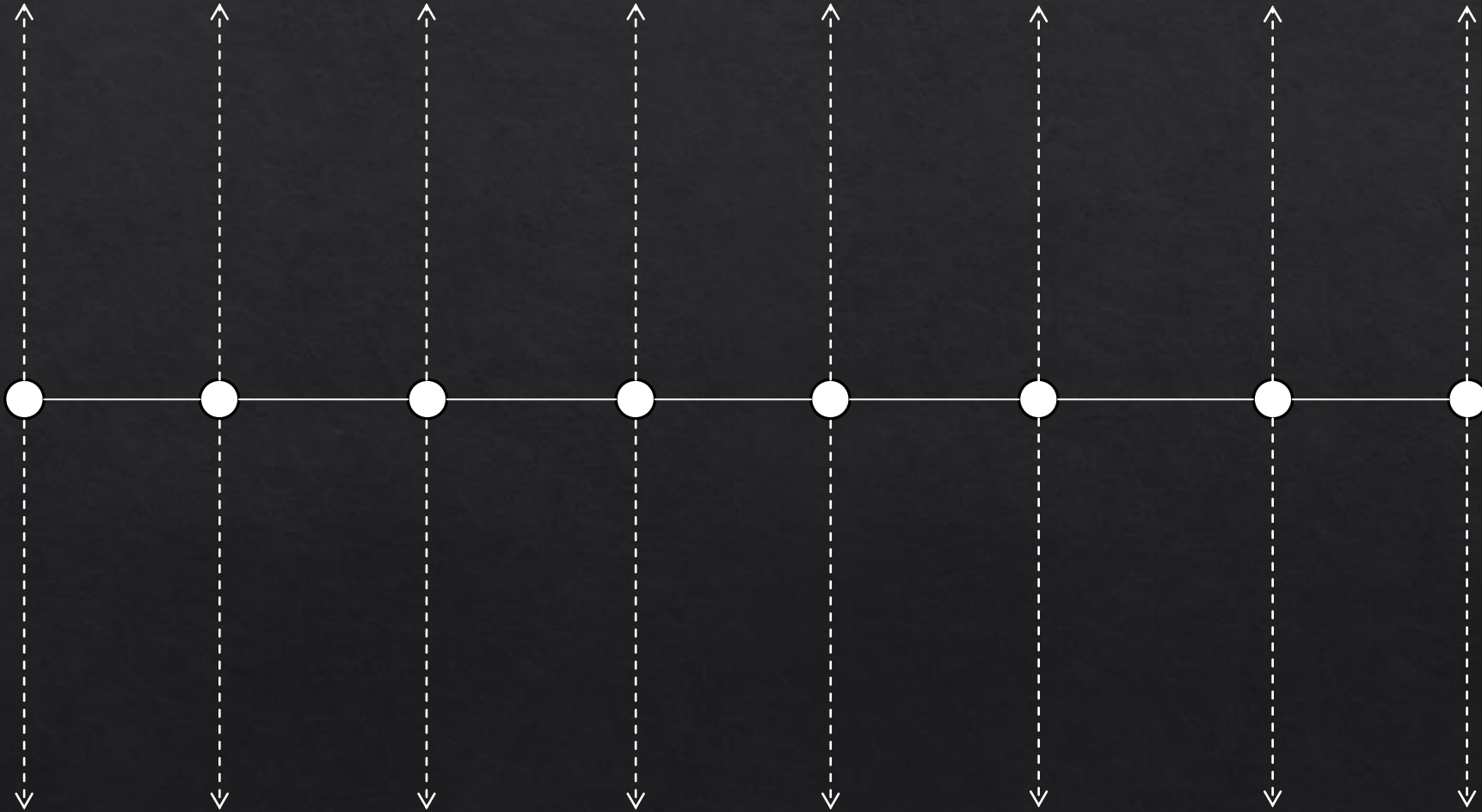
버텍스 셰이더 사용 애니메이션



위 Lines 를 물결치는 것과 같이 움직여 보자

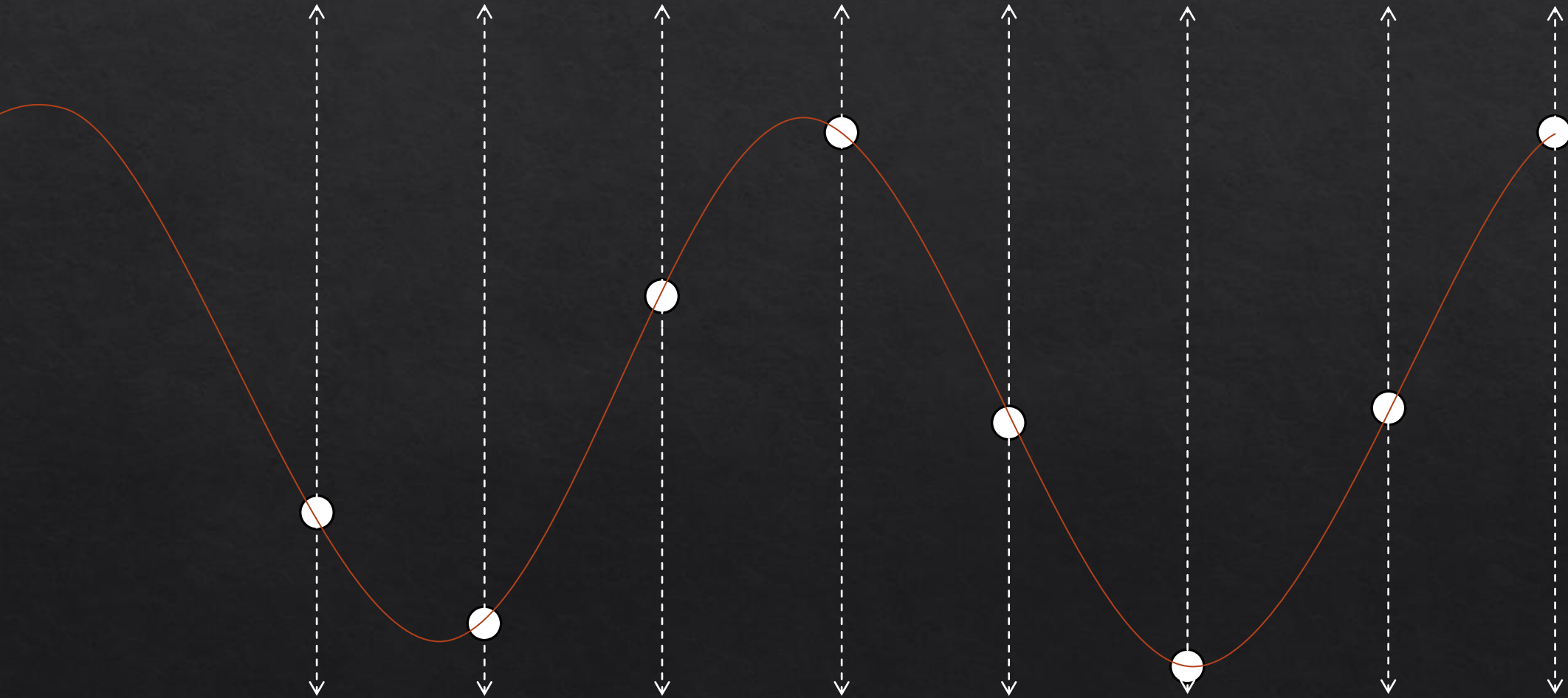
VBO Line

버텍스 셰이더 사용 애니메이션



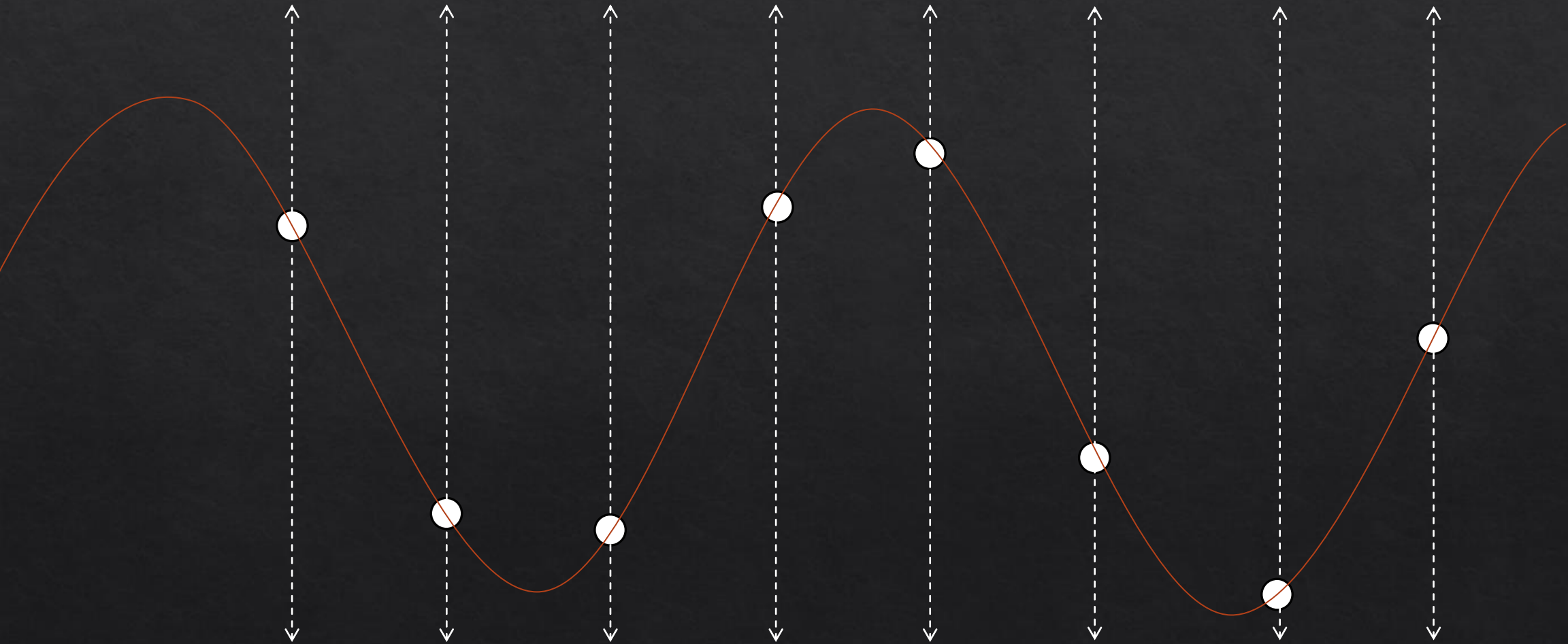
버텍스 셰이더 사용 애니메이션

Time == 0



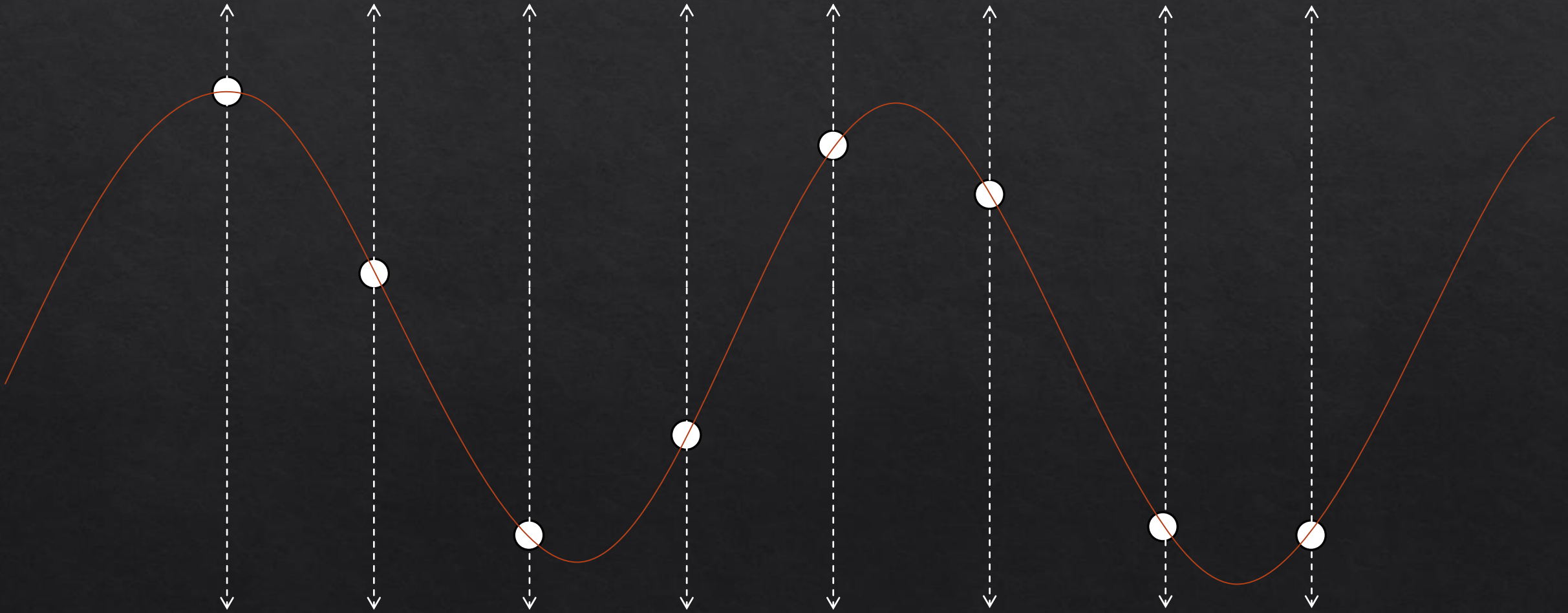
버텍스 셰이더 사용 애니메이션

Time == 0.1



버텍스 셰이더 사용 애니메이션

Time == 0.2



버텍스 셰이더 사용 애니메이션

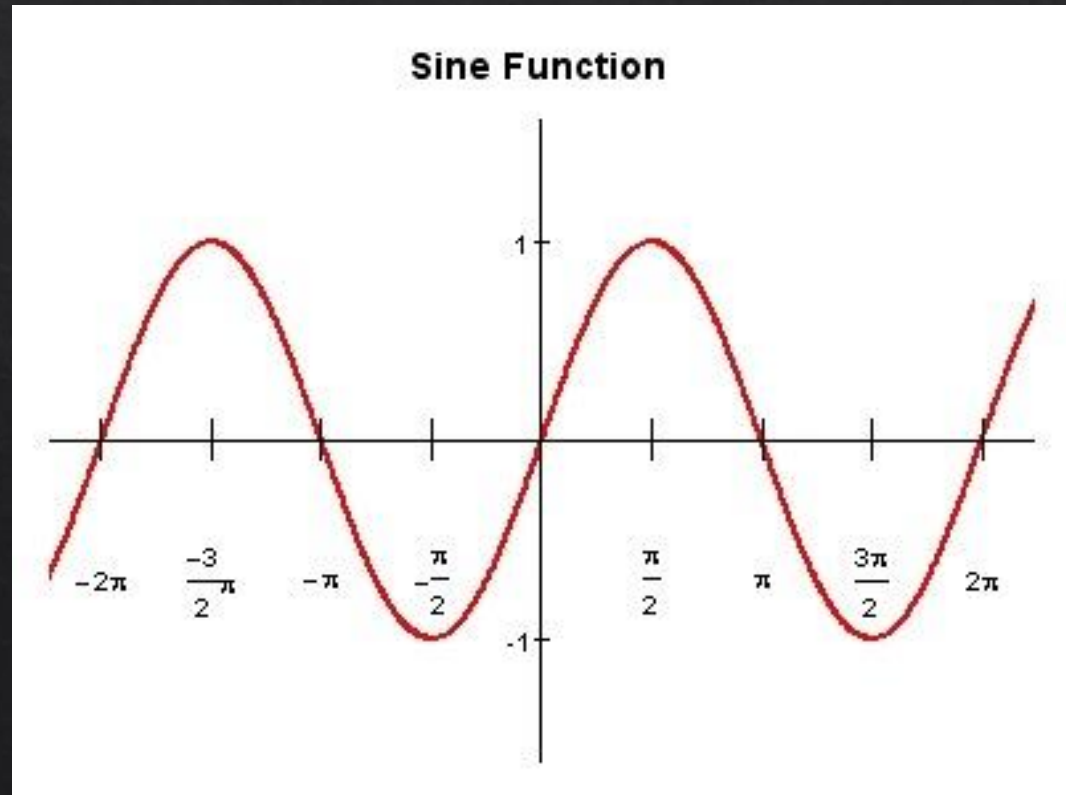
- ◇ 즉, 버텍스의 Y 좌표를 위아래로 움직여주는 셰이더 프로그램이 필요함
- ◇ $\sin(\text{radian})$ 함수 이용

버텍스 셰이더 사용 애니메이션

- ◇ 버텍스 하나만 있다고 가정해보고 위 아래로 움직여 보자



$(0.f, \sin(\text{radian}), 0.f)$



버텍스 셰이더 사용 애니메이션

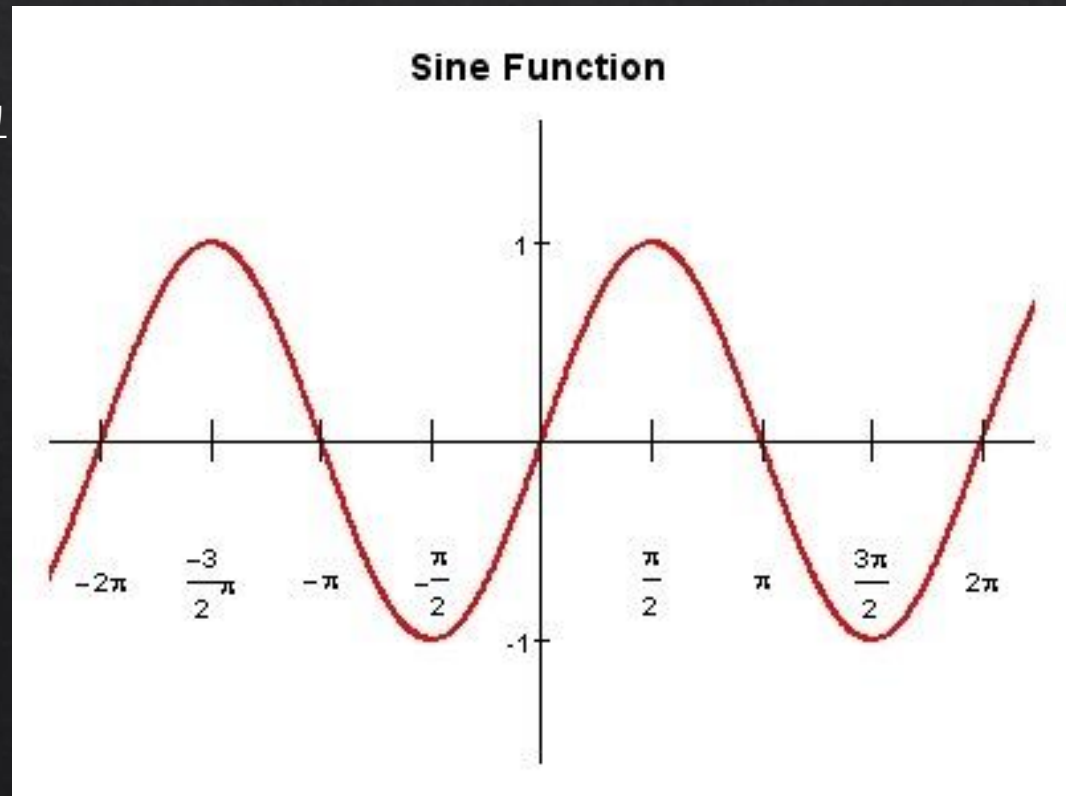
- ◇ Sin 함수 입력 값을 변화 시키면 버텍스가 위 아래로 움직이게 됨

Time을 uniform 으로 넣어주고
증가하는 값을 넣어주면
애니메이션 가능!

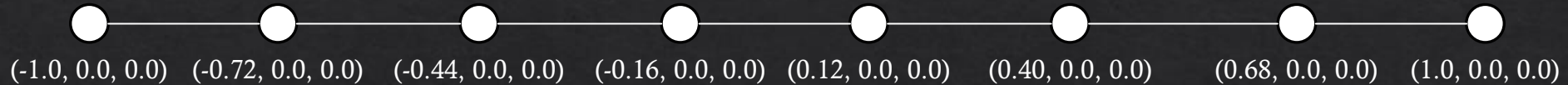


$(0.f, \sin(\text{time}), 0.f)$

uniform float time;



버텍스 셰이더 사용 애니메이션



하지만 모든 버텍스에 대해 (... , sin(time), ...) 와 같이 값을 주면 모든 버텍스가 동일한 높이를 가지게 되어 모두 위 아래로 같이 움직이게 됨

이를 해결하기 위해선 sin함수 입력 값이 모두 달라야 함

입력된 time 값에 버텍스의 x 위치를 더해보자.

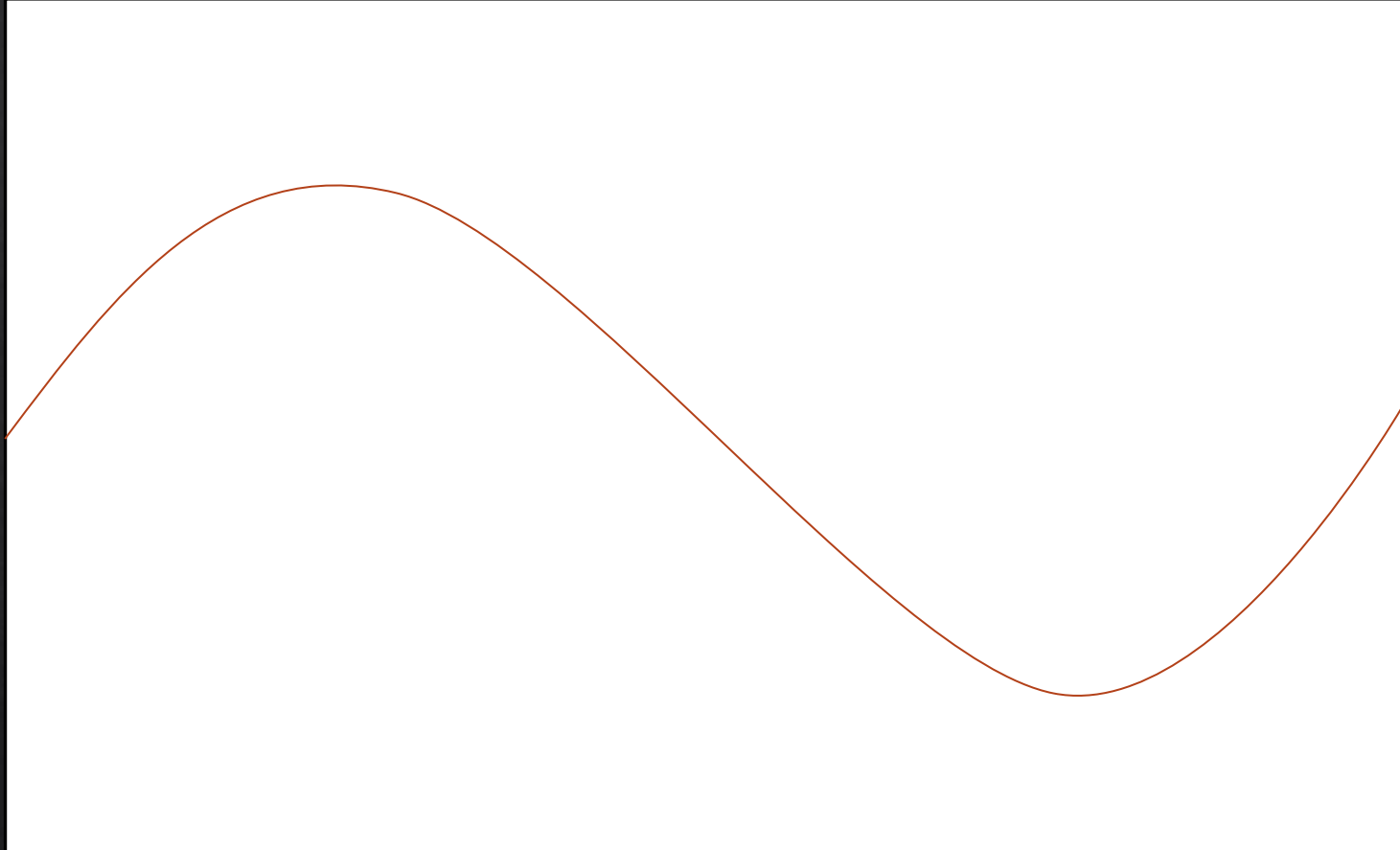
```
- float newRadian = Position.x*PI + time;
```

시작 높이가 다른 상태로 위 아래로 움직여 물결치는 듯한 효과를 냄

실습

프로그래먼트 셰이더 사용 애니메이션

프로그래밍 셰이더 사용 애니메이션



버텍스 셰이더 애니메이션과 마찬가지로 물결치도록 만들어 보자

프로그래먼트 셰이더 사용 애니메이션

$(-1.0, 1.0, 0.0)$

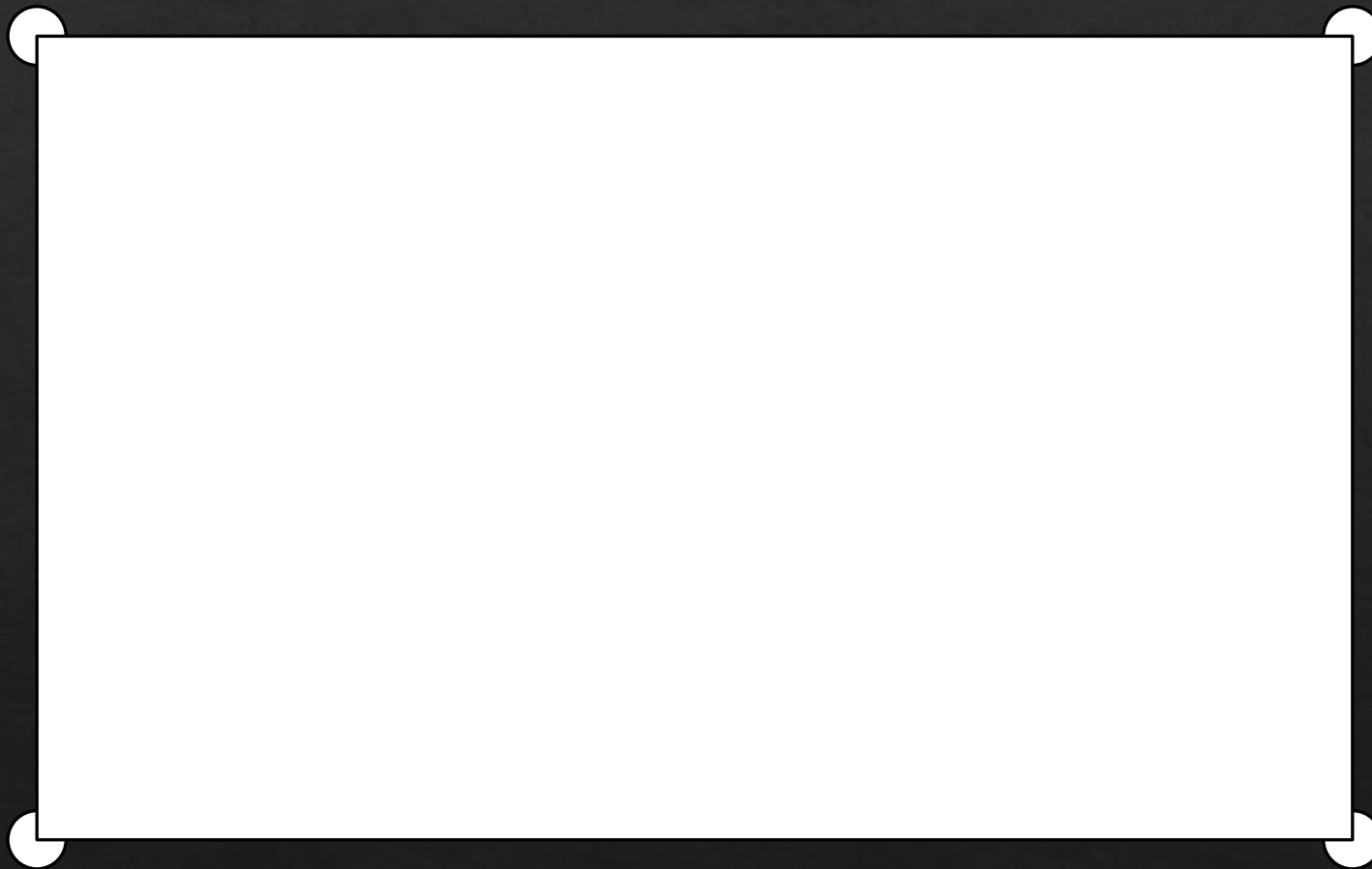
$(1.0, 1.0, 0.0)$

$(-1.0, -1.0, 0.0)$

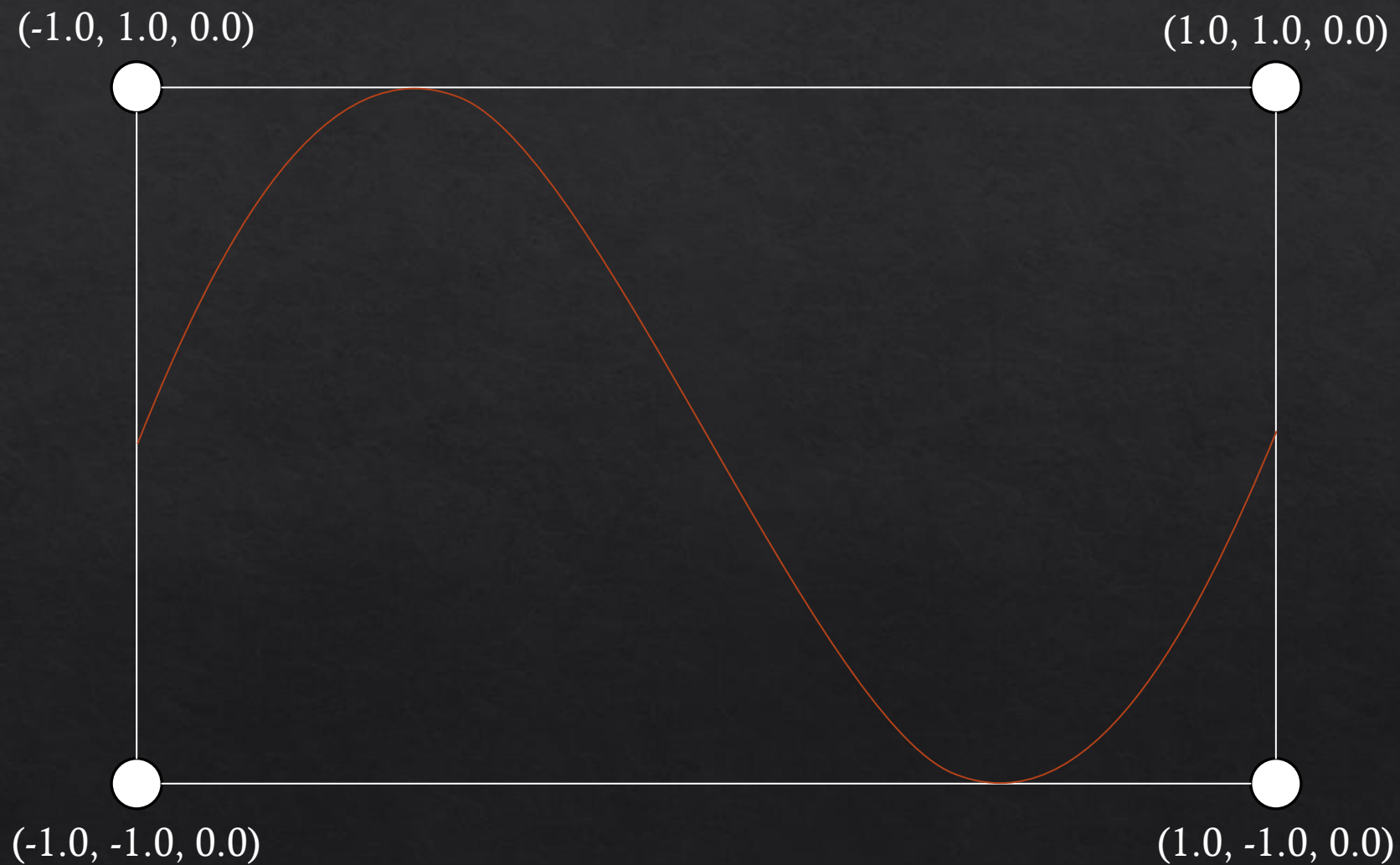
$(1.0, -1.0, 0.0)$



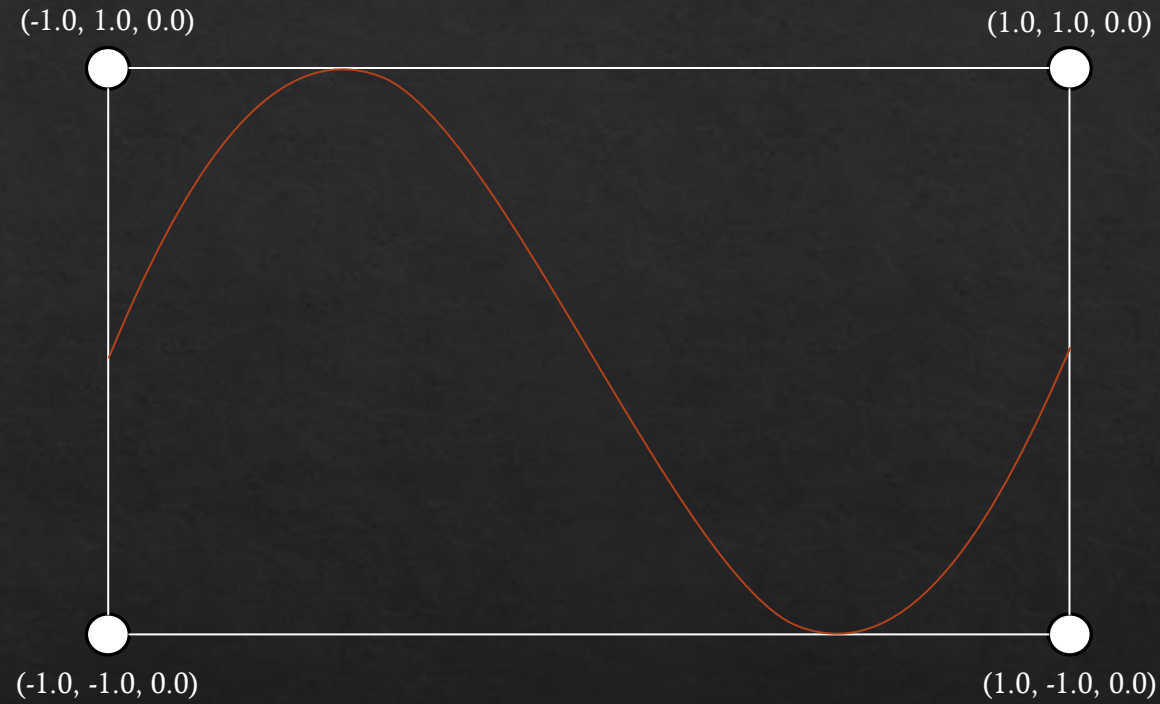
프로그래먼트 셰이더 사용 애니메이션



프로그래먼트 셰이더 사용 애니메이션

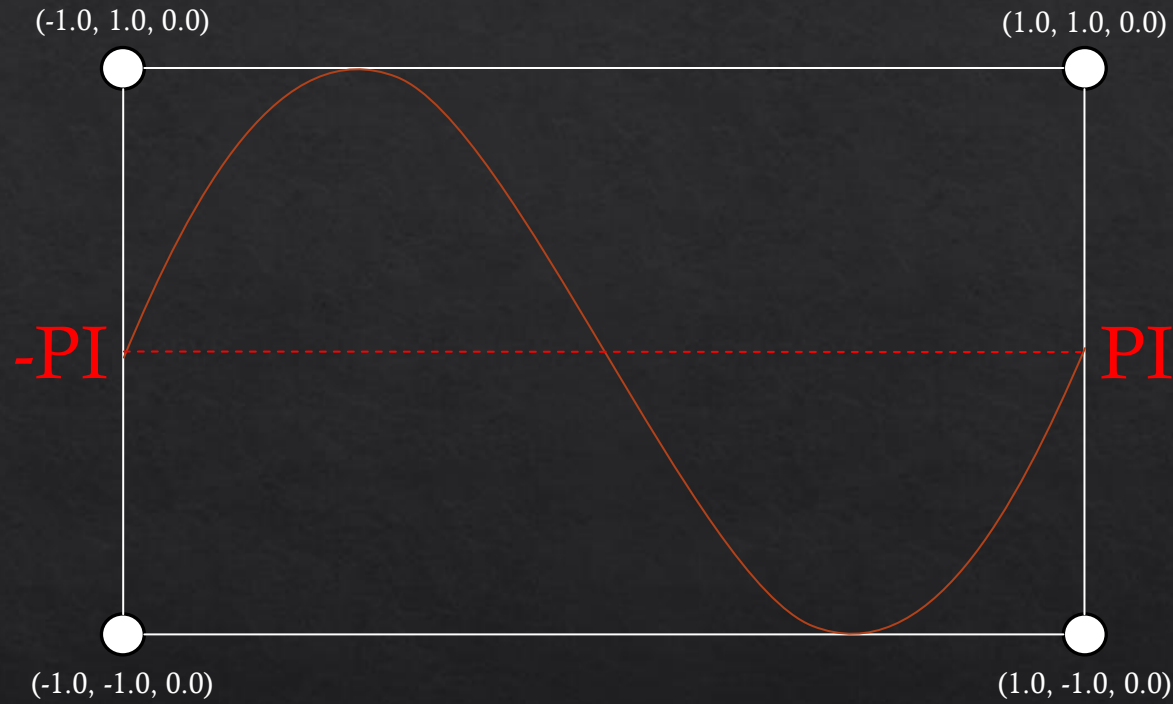


프로그래먼트 셰이더 사용 애니메이션



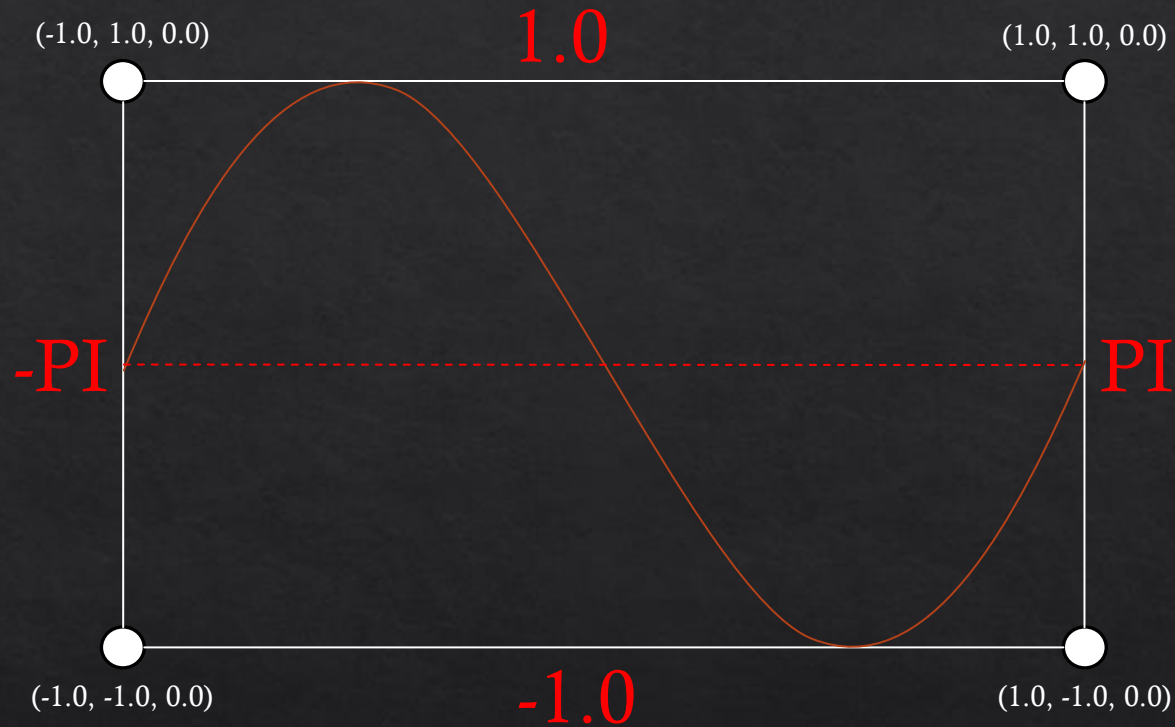
버텍스가 4개밖에 없기 때문에 버텍스 애니메이션으로는 불가능
→ 프로그래먼트 셰이더에서 $\sin(\text{radian})$ 함수를 사용!

프로그래먼트 셰이더 사용 애니메이션



프로그래먼트 셰이더의 입력으로 $-PI \sim PI$ 의 입력이 들어와야
sin 함수 표현이 가능함

프로그래먼트 셰이더 사용 애니메이션



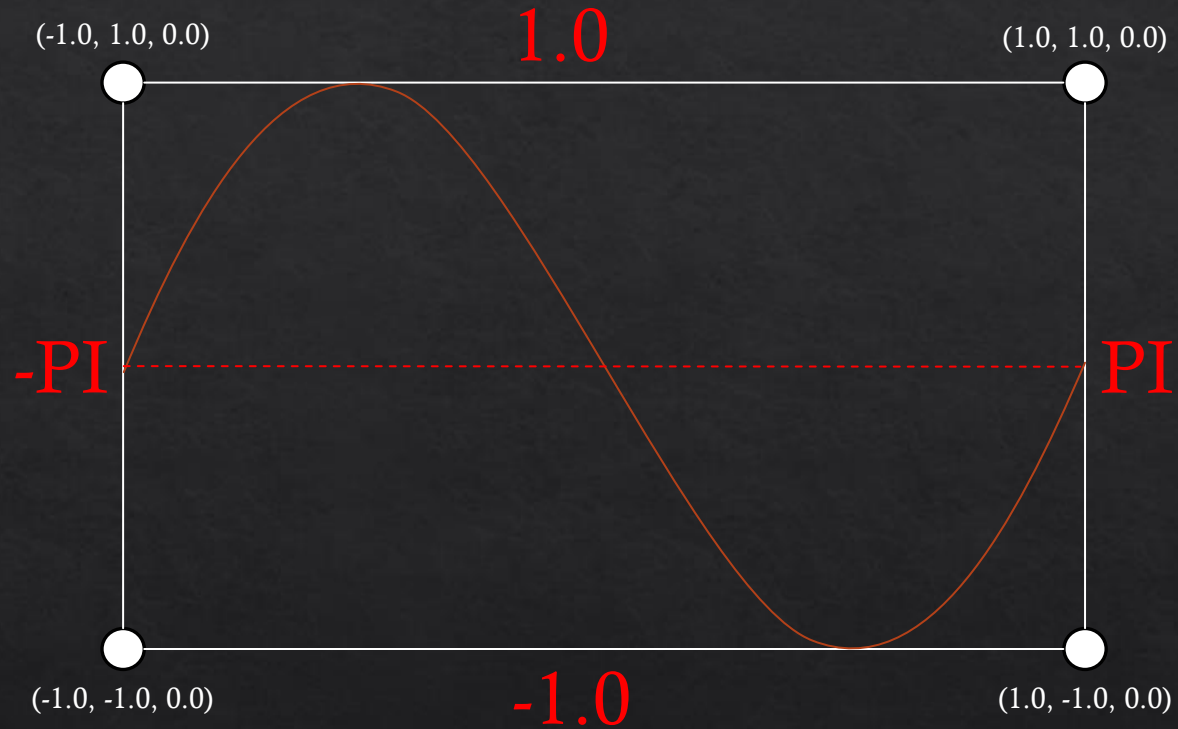
또한 y 축 좌표가 있어야만 sin 함수 결과와 비교하여
프로그래먼트 색을 정할 수 있음

프로그래밍 셰이더 사용 애니메이션

vertex shader output

`out vec2 vPos;`

X값엔 $-\pi \sim \pi$ 가
넘어가도록 값을 넣으며
y값엔 $-1.0 \sim 1.0$ 이
넘어가도록 값을 넣는다.

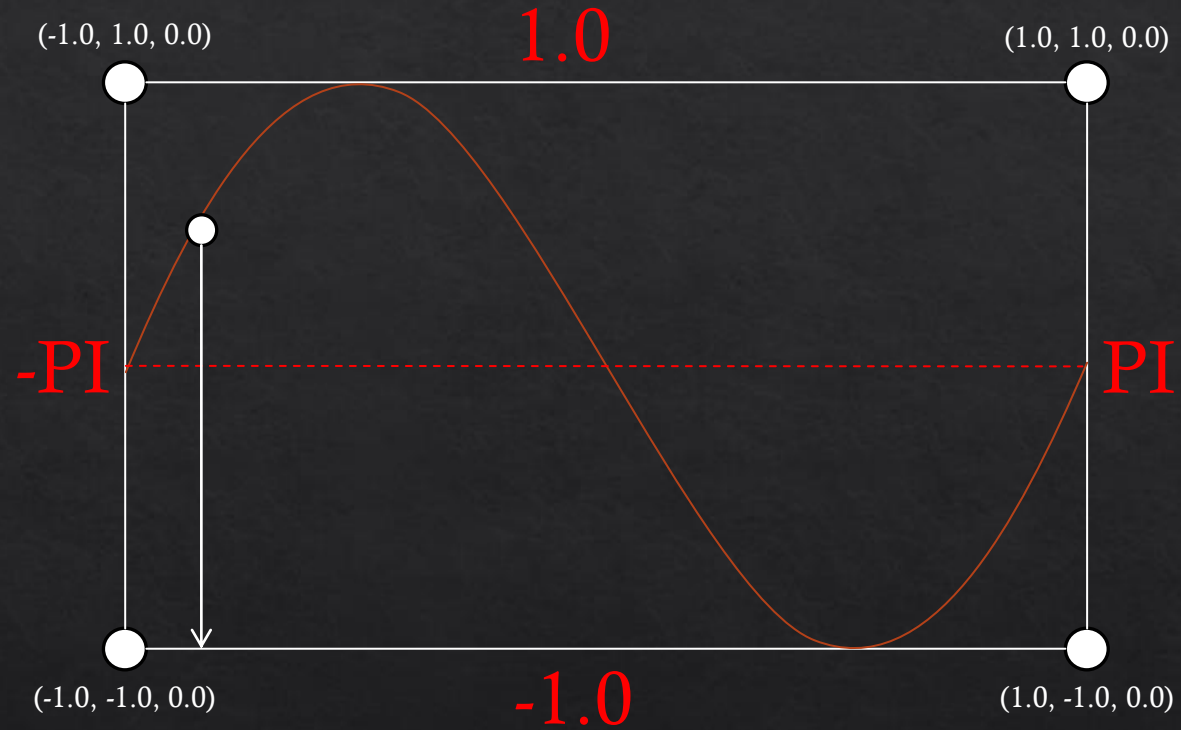


프래그먼트 셰이더 사용 애니메이션

fragment shader input

In vec2 vPos;

```
float y = sin(vPos.x);  
if(vPos.y < y)  
....
```



실습

실습

◆ 지난 시간 코드에 이어서 구현