🎮

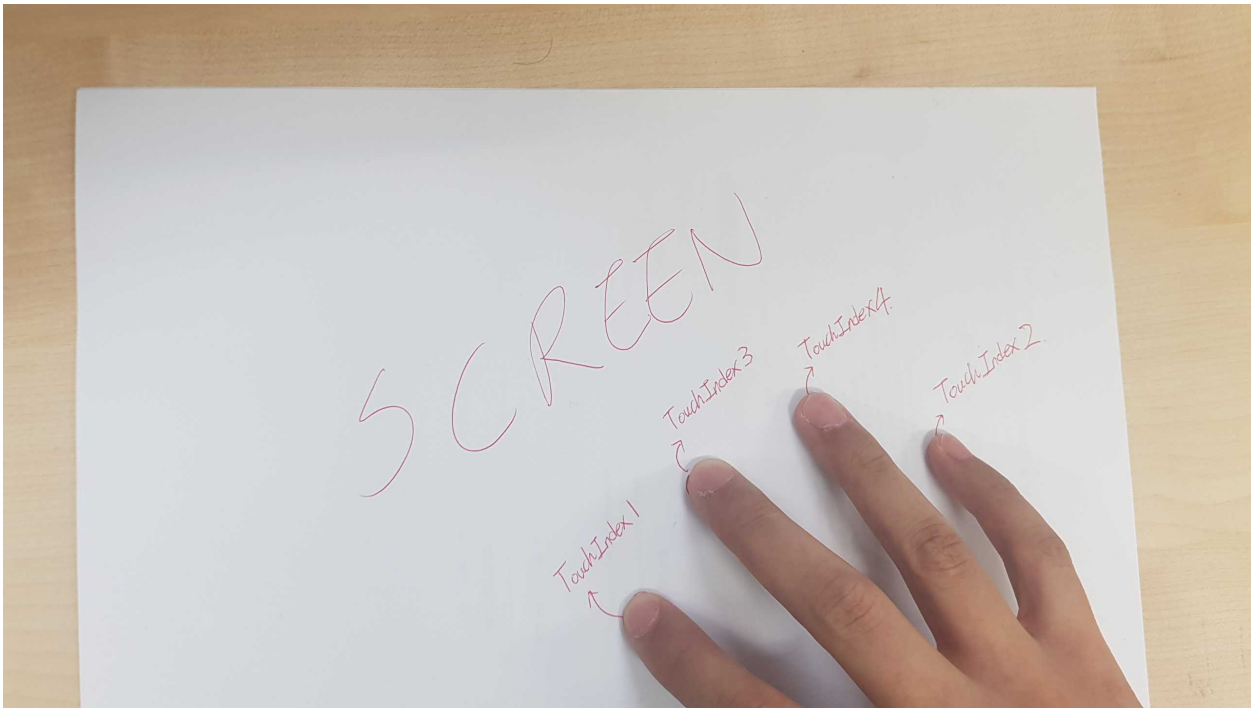
UE 4.27 터치 입력 제어(컨트롤러 회전)

Created	@January 17, 2022 10:28 AM
Tags	

- 본문에서는 스크린에 터치하고 스와이프하여 컨트롤러를 회전시킵니다.
- 본문에서는 TouchIndex의 정의를 내리고 터치가 이루어지면 어떤 일이 일어나는지, 그리고 현재 터치 상태는 어떻게 확인하는지를 포함합니다.

What is TouchIndex?

- TouchIndex란 스크린에 터치를 구분하기 위해 Index를 부여하여 구분하는 걸 말합니다.
- TouchIndex는 스크린에 터치된 순서를 기준으로 정해집니다.
- 다음 사진을 참고하여 주십시오.



- 위 사진의 터치 인덱스는 다음의 순서로 터치되었고 그에 따라 TouchIndex가 정해집니다.
  1. 검지로 스크린을 터치합니다. 검지가 터치한 정보를 가리키는 인덱스는 TouchIndex1이 됩니다.
  2. 소지로 스크린을 터치합니다. 소지가 터치한 정보를 가리키는 인덱스는 TouchIndex2가 됩니다.
  3. 중지로 스크린을 터치합니다. 중지가 터치한 정보를 가리키는 인덱스는 TouchIndex3이 됩니다.
  4. 약지로 스크린을 터치합니다. 약지가 터치한 정보를 가리키는 인덱스는 TouchIndex4가 됩니다.
  5. 이 상태에서 만약 중지와 약지를 스크린에서 떼고 약지를 다시 터치하면 약지가 TouchIndex3이 됩니다.

- APlayerController에서는 터치가 이루어지면 bool InputTouch() 함수가 호출되고 해당 TouchIndex의 정보를 저장할 수 있습니다.

```
virtual bool InputTouch(uint32 Handle, ETouchType::Type Type, const FVector2D& TouchLocation, float Force, FDateTime DeviceTimestamp, uint3

// Handle          - 터치된 TouchIndex입니다.
// Type            - 터치 타입을 가리킵니다.
//                   터치타입은
//                   Began,
//                   Moved,
//                   Stationary,
//                   ForceChanged,
//                   FirstMove,
//                   Ended로 정의되어 있습니다.
// TouchLocation   - 터치가 이루어진 스크린상의 위치입니다.
// Force           - 터치된 힘을 가리키는 걸로 생각됩니다. 정확하게는 알 수 없음.
// DeviceTimestamp - 터치가 이루어진 시간을 디바이스의 시간을 기준으로 나타냅니다.
// TouchPadIndex   - 현재 알 수 없음.
```

- bool APlayerController::InputTouch()에서 터치 인덱스별로 정보를 다음과 같이 갱신할 수 있습니다.

```
// AMyPlayerController.h

/** Touch의 정보를 담는 구조체입니다. 타입과 위치를 갱신합니다. */
USTRUCT()
struct FTouchIndexInfo
{
  GENERATED_BODY()

  FTouchIndexInfo()
    : Type(ETouchType::Ended)
    , LastCoord(0.f)
    , CurCoord(0.f)
  {}

  /** Touch type of Current Frame. */
  ETouchType::Type Type;

  /** Last Coordinate of touch. */
  FVector2D LastCoord;

  /** Current Coordinate of touch. */
  FVector2D CurCoord;
};

UCLASS()
class TOUCHINTERFACE_API AMyPlayerController : public APlayerController
{
  GENERATED_BODY()

  TArray<FTouchIndexInfo> TouchIndexInfos;

public:
  AMyPlayerController();

  virtual bool InputTouch(uint32 Handle, ETouchType::Type Type, const FVector2D& TouchLocation, float Force, FDateTime DeviceTimestamp, uin

  // ... 기타 생략 ...
};
```

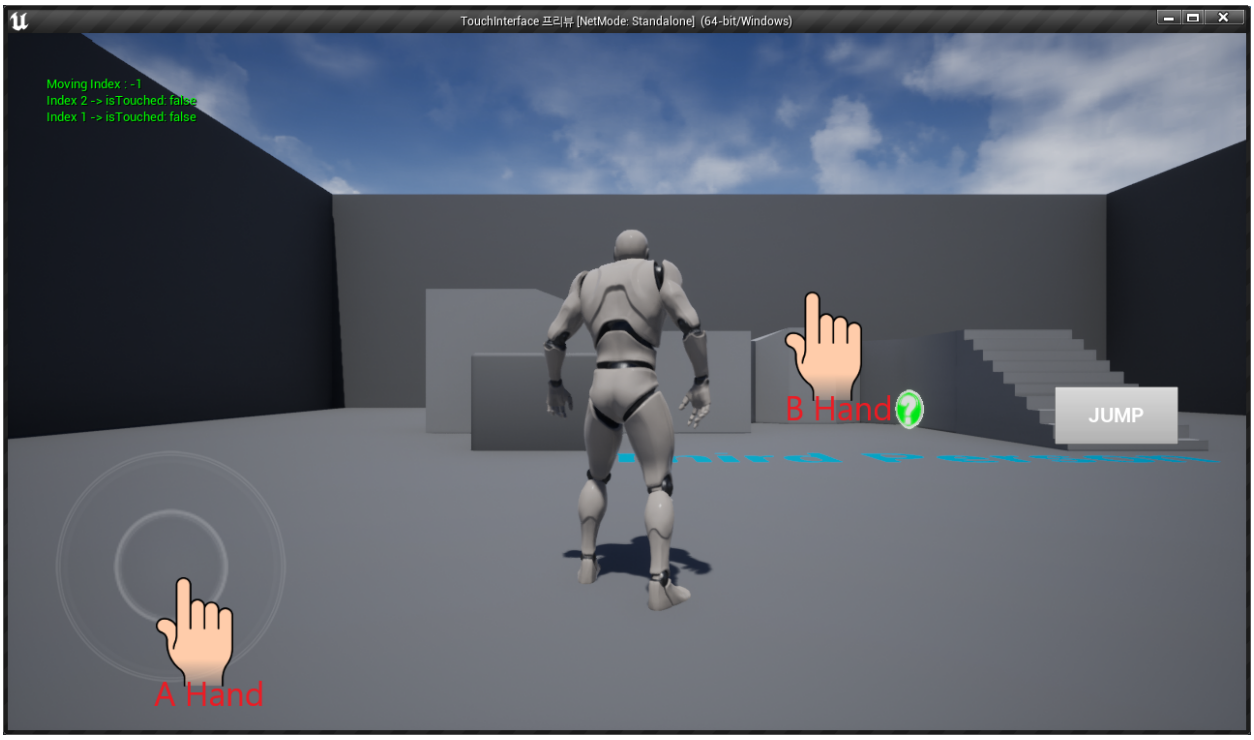
```
// AMyPlayerController.cpp

// TouchIndexInfos 초기화 생략...

bool AMyPlayerController::InputTouch(uint32 Handle, ETouchType::Type Type, const FVector2D& TouchLocation, float Force,
                                     FDateTime DeviceTimestamp, uint32 TouchpadIndex)
{
  TouchIndexInfos[Handle].Type= Type;
  TouchIndexInfos[Handle].CurCoord= TouchLocation;
```

```
  return Super::InputTouch(Handle, Type, TouchLocation, Force, DeviceTimestamp, TouchpadIndex);
}
```

- 위의 코드대로라면 손가락은 총 10개이기 때문에 총 10개의 TouchIndexInfo가 업데이트될 것입니다.
- 이제 TouchIndexInfos를 이용하여 현재의 터치 상태를 알 수 있고 위치를 갱신하여 컨트롤러를 Rotate 시킬 수 있게 됩니다.
- **APlayerController::GetInputTouchState()**로도 현재 프레임의 터치 정보를 가져올 수 있습니다.

회전 로직

- 회전은 스크린에 터치된 손가락이 하나인 경우에만 허용하기로 하였습니다.
- 스크린에 터치된 손가락의 현재 위치와 직전 프레임의 위치를 빼서 그 거리를 기준으로 컨트롤러를 회전시킵니다.
- TouchIndex의 정보는 TouchInterface를 사용하는 경우에는 Consume되고 스크린에 터치되는 순서에 따라 TouchIndex가 정해지기 때문에 컨트롤러를 회전시키는 TouchIndex는 매번 달라질 수 있습니다. 아래에 자세하게 서술해 놓았습니다.



- 손은 양손을 모두 쓴다고 가정, 왼손을 Hand A, 오른손을 Hand B라고 하겠습니다.
- Hand A의 한 손가락으로 TouchInterface를 움직여서 캐릭터를 조종하고 Hand B의 한 손가락으로 스크린을 터치하고 스와이프하여 컨트롤러를 회전시키게 됩니다.
- Hand A와 Hand B의 터치 순서에 따라 Hand B의 검지의 TouchIndex가 달라집니다.
  1. A를 먼저 터치하고 B를 터치하는 경우
     - A의 손가락이 TouchIndex1이 되고 TouchIndex1의 입력이 Consume됩니다. 즉, TouchIndex1의 입력은 업데이트되지 않습니다.
     - B의 손가락은 TouchIndex2가 되고 업데이트가 됩니다.
  2. B를 먼저 터치하고 A를 터치하는 경우
     - B의 손가락이 TouchIndex1이 되고 업데이트됩니다.

    - A의 손가락은 TouchIndex2가 되고 TouchIndex2의 입력이 Consume됩니다.
  3. A의 손가락은 터치하지 않고 B의 손가락만 터치되는 경우
    - B의 손가락이 TouchIndex1이 되고 업데이트 됩니다.
- 위와 같이 B의 손가락의 TouchIndex가 정해지기 때문에 MovingTouchIndex, 즉 회전시킬 TouchIndex는 다음과 정해집니다.
  - TouchIndex1의 입력이 있고 TouchIndex2의 입력이 없는 경우 TouchIndex1로 회전
  - TouchIndex1의 입력이 없고 TouchIndex2의 입력이 있는 경우 TouchIndex2로 회전
- 회전시킬 TouchIndex가 정해지면 그 TouchIndex의 현재 좌표와 직전 프레임에서의 좌표의 거리 차이를 통해 회전시킵니다.
- 터치가 움직이기 시작할 때에는 직전 프레임의 좌표인 LastCoord와 현재 프레임의 좌표인 CurCoord를 같게 만들어 주고, 터치가 움직이면 CurCoord - LastCoord를 시킵니다. 이 값을 DeltaCoord라고 하고 이 순간변화된 값으로 APlayerController의 RotationInput을 업데이트해줍니다. APlayerController의 PlayerTick()에서 RotaionInput을 이용해서 Rotation을 업데이트하게됩니다.

FTouchRotator

- 터치를 이용한 플레이어 컨트롤러를 회전시키는 클래스입니다.

```
// FTouchRotator.h

#define INVALID_TOUCH_INDEX (-1)
#define MAX_ROTATABLE_TOUCH_INDEX (2)      // TouchIndex3 이상으로는 회전 시키지 않습니다.

class APlayerController;

class TOUCHINTERFACE_API FTouchRotator
{
  /** TouchIndex들의 정보를 담고 있는 배열. */
  TArray<FTouchIndexInfo> TouchIndexInfos;

  /** PlayerController를 가리키는 약포인터. */
  TWeakObjectPtr<APlayerController> OwnerPC;

public:
  FTouchRotator();
  FTouchRotator(const FTouchRotator& Other) = delete;
  FTouchRotator& operator=(const FTouchRotator& Rhs) = delete;
  ~FTouchRotator() = default;

  FORCEINLINE void SetOwnerPlayerController(APlayerController* InOwnerPC) { OwnerPC = InOwnerPC; }

  /** APlayerController::InputTouch()에서 호출됩니다. 터치 타입과 좌표를 저장합니다. */
  void SetTouchIndexInfo(int32 TouchIndex, ETouchType::Type TouchType, const FVector2D& NewTouchCoord);

  /** APlayerController::PlayerTick()에서 호출됩니다. */
  void RotateController(float CameraTurnRate, float CameraLookUpRate, float Deltatime);

private:
  /** 현재 프레임에서 TouchIndex를 넣어주면 그 TouchIndex의 TouchType에 따라 Touch가 되었는지 반환합니다. */
  bool IsTouched(ETouchIndex::Type TouchIndex);

  /** 컨트롤러를 회전시킬 TouchIndex를 선택합니다. TouchIndex1 혹은 TouchIndex2가 됩니다. */
  int32 PickTouchIndex();
};
```

```
// FTouchRotator.cpp

FTouchRotator::FTouchRotator()
{
  // TouchIndexInfos는 두개만 사용될 것입니다.
  // TouchIndexInfos를 초기화합니다.
  TouchIndexInfos.Init(FTouchIndexInfo(), MAX_ROTATABLE_TOUCH_INDEX);
}

void FTouchRotator::SetTouchIndexInfo(int32 TouchIndex, ETouchType::Type TouchType, const FVector2D& NewTouchCoord)
{
  if (TouchIndex > ETouchIndex::Touch2)
  {
    // TouchIndex 3 이상의 정보는 저장하지 않습니다.
```

```
    return;
  }

  TouchIndexInfos[TouchIndex].Type = TouchType;
  TouchIndexInfos[TouchIndex].CurCoord = NewTouchCoord;

  if (TouchType == ETouchType::Began)
  {
    // 터치가 시작되었을 때는 LastCoord와 CurCoord를 같게 합니다.
    TouchIndexInfos[TouchIndex].LastCoord = TouchIndexInfos[TouchIndex].CurCoord;
  }
}

void FTouchRotator::RotateController(float CameraTurnRate, float CameraLookUpRate, float Deltatime)
{
  if (!OwnerPC.IsValid())
  {
    return;
  }

  // 컨트롤러를 회전시킬 TouchIndex는 매 프레임마다 달라질 수 있습니다.
  // 그렇기 때문에 매 틱마다 회전시킬 TouchIndex를 갱신합니다.
  const int32 MovingIndex = PickTouchIndex();

  if (MovingIndex != INVALID_TOUCH_INDEX)
  {
    const FVector2D DeltaDistance = TouchIndexInfos[MovingIndex].CurCoord - TouchIndexInfos[MovingIndex].LastCoord;

    OwnerPC.Get()->RotationInput.Pitch -= DeltaDistance.Y * CameraLookUpRate * Deltatime;
    OwnerPC.Get()->RotationInput.Yaw += DeltaDistance.X * CameraTurnRate * Deltatime;
  }

  for (int32 i = 0; i < ETouchIndex::Touch3; ++i)
  {
    TouchIndexInfos[i].LastCoord = TouchIndexInfos[i].CurCoord;
  }
}

bool FTouchRotator::IsTouched(ETouchIndex::Type TouchIndex)
{
  return TouchIndexInfos[TouchIndex].Type != ETouchType::Ended;
}

int32 FTouchRotator::PickTouchIndex()
{
  // TouchIndex가 유효한 인덱스가 아닌 경우에는 회전시키지 않습니다.
  int32 _result = INVALID_TOUCH_INDEX;

  if (IsTouched(ETouchIndex::Touch1) && !IsTouched(ETouchIndex::Touch2))
  {
    _result = ETouchIndex::Touch1;
  }
  else if (!IsTouched(ETouchIndex::Touch1) && IsTouched(ETouchIndex::Touch2))
  {
    _result = ETouchIndex::Touch2;
  }

  return _result;
}
```

AMyPlayerController

```
// AMyPlayerController.h

UCLASS()
class TOUCHINTERFACE_API AMyPlayerController : public APlayerController
{
  GENERATED_BODY()

  UPROPERTY(EditDefaultsOnly, meta=(AllowPrivateAccess="true"))
  float CameraTurnRate;

  UPROPERTY(EditDefaultsOnly, meta=(AllowPrivateAccess="true"))
  float CameraLookUpRate;
```

```
  // TouchRotator를 컴포넌트로 가지고 있습니다.
  TUniquePtr<FTouchRotator> TouchRotator;

public:
  AMyPlayerController();

  virtual bool InputTouch(uint32 Handle, ETouchType::Type Type, const FVector2D& TouchLocation, float Force, FDateTime DeviceTimestamp, uin

  virtual void PlayerTick(float DeltaTime) override;
};
```

```
// AMyPlayerController.cpp

AMyPlayerController::AMyPlayerController()
  : Super()
    , CameraTurnRate(20.f)
    , CameraLookUpRate(20.f)
{
  // TouchRotator를 생성, 초기화합니다.
  TouchRotator = MakeUnique<FTouchRotator>();
  TouchRotator->SetOwnerPlayerController(this);
}

bool AMyPlayerController::InputTouch(uint32 Handle, ETouchType::Type Type, const FVector2D& TouchLocation, float Force,
                                     FDateTime DeviceTimestamp, uint32 TouchpadIndex)
{
  TouchRotator->SetTouchIndexInfo(Handle, Type, TouchLocation);

  return Super::InputTouch(Handle, Type, TouchLocation, Force, DeviceTimestamp, TouchpadIndex);
}

void AMyPlayerController::PlayerTick(float DeltaTime)
{

  TouchRotator->RotateController(CameraTurnRate, CameraLookUpRate, DeltaTime);

  // APlayerController::PlayerTick()에서는 RotationInput으로 현재의 Rotation을 Update합니다.
  // 그렇기 때문에 먼저 RotationInput을 갱신하고 Super::PlayerTick(float)를 호출합니다.
  Super::PlayerTick(DeltaTime);
}
```