MediaPipe 및 Unreal Engine 기반 모션 인식 댄스 게임

Team

1팀

Instructor

박능수 교수님

Team Members

201711836 송호영

201710177 정은호

201811652 손하빈

목차

1. 개요

1.1 소개

댄스 게임은 음악, 운동 그리고 재미있는 게임 요소가 결합된 창의적인 엔터테인먼트 로서 많은 사람들에게 사랑받고 있습니다. 주로 콘솔 게임에서 큰 인기를 얻으며, 컨트롤러를 통해 사용자의 움직임을 인식하고, 사용자로 하여금 음악에 맞춰 정확한 시간에 정확한 동작을 수행하도록 하는 독특한 플레이 방식을 제공하고 있습니다.

콘솔 및 컨트롤러 기반 댄스 게임은 별도의 기기를 필요로 하며, 컨트롤러만으로 사용자의 몸 전체 움직임을 인식하고자 하기 때문에 댄스 동작 인식에 한계를 갖고 있습니다. 이 프로젝트에서는 더욱 복잡하고 현실적인 움직임 인식이 가능한 댄스 게임을 개발하고자 합니다.

실시간으로 사용자의 실제 움직임을 인식하고 판단하는 댄스 게임을 통해 사용자에게 인터랙티브한 경험을 제공하고, 사용자의신체 활동을 증진하는 데 도움을 줄 수 있을 것으로 기대합니다.

1.2 목표

- 사용자가 카메라를 켜고 음악에 맞춰 춤을 추면 정확도에 따라 점수를 부여하는 방식으로 구현합니다.
- 한 곡으로 이루어진 하나의 스테이지로 완성합니다.

1.3 용어

이 문서에서 사용되는 용어들의 의미를 약속합니다. 여기서 약속하는 용어들은 다음과 같습니다.

- **포즈**: 연속적인 댄스 영상에서 특정 프레임에서 캐릭터의 특정 동작
- **Target 포즈**: 사용자가 따라해야 할 대상 동작
- **User 포즈**: 카메라를 통해 인식된 사용자의 실시간 동작
- **점수**: Target 포즈 벡터 및 User 포즈 간 유사도
- **등급**: 점수를 기준으로 최종 산출되는 게임의 결과 (A, B, C)

2. 기본사항

2.1 개발 환경

- **Unreal Engine** : 이 프로젝트에서 핵심으로 사용한 게임 엔진으로, 게임의 시각적 요소와 물리적 시뮬레이션을 담당합니다. 사용자의 움직임을 게임 캐릭터에 적용하기 위해 IK와 FABRIK 시스템을 함께 사용합니다.
- **C++** : Unreal Engine에서 사용되는 주된 프로그래밍 언어로, 게임의 로직과 애니메이션, 물리 시뮬레이션 등을 구현하는 데 사용합니다.
- **Python** : MediaPipe 구동을 위한 코드 작성, 즉 사용자의 움직임 인식 및 분석, 그리고 이에 대한 데이터 처리에 사용됩니다. Unreal Engine과 MediaPipe 간 소켓 통신을 위한 코드 작성에서도 사용합니다.
- **GitHub** : 소스 코드 관리 시스템을 통해 프로젝트 버전을 관리합니다.
- **Notion** : 개발 일정, 할당된 역할, 협업을 위한 문서 등을 관리하고 공유하는 데에 사용합니다.

2.2 프로그램 실행

- **MediaPipe 실행** : 파이썬으로 작성된 MediaPipe를 실행합니다. MediaPipe는 소켓 통신을 위한 서버로 동작하므로 Unreal Engine이 실행되기 전 실행되고 있어야 합니다.
- **Unreal Engine 게임 실행** : Unreal Engine으로 만든 게임을 이후 실행합니다. 게임은 MediaPipe와 소켓 통신에서 클라이언트로 동작하며 사용자의 움직임 데이터를 받아들이고, 해당 데이터에 따라 게임 캐릭터 움직임을 조정합니다.

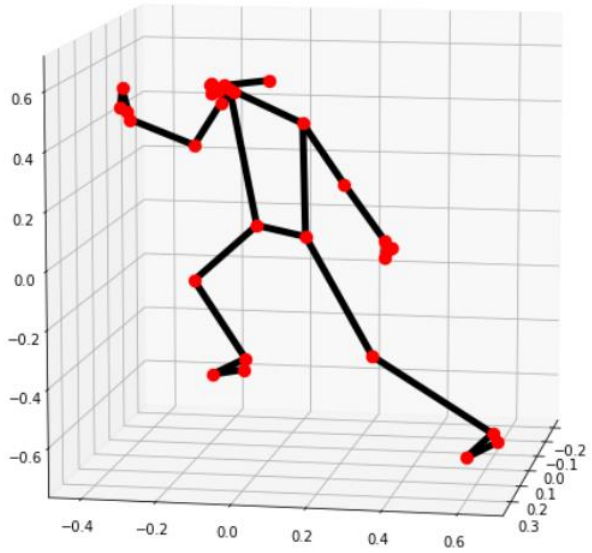
MediaPipe는 실시간으로 사용자의 움직임 데이터를 생성하고, Unreal Engine은

이 데이터를 받아 게임 환경에서 사용자의 움직임을 시각화합니다. Unreal Engine 동작은 MediaPipe 동작에 의존하므로, MediaPipe 동작이 중단되면 영향을 받게 됩니다.

3. 사용 기술

3.1 MediaPipe

Google에서 개발한 오픈소스 소프트웨어로서, 실시간 주요 관절 위치 검출 및 추출을 포함한 컴퓨터 비전의 다양한 기능을 제공합니다. *이 프로젝트에서는* Python을 통해 구현되었습니다.



많은 공통점을 공유하는 OpenPose 라이브러리 또한 사용 고려 대상이었으나, 빠르고 캐주얼하게 즐기는 댄스 게임의 특성과 개발의 용이함을 고려하여 속도와 사용성이 높은 Mediapipe로 결정하였습니다.

	MediaPipe	OpenPose
공통점	머신 러닝 및 컴퓨터 비전을 활용한 실시간 인체 동작 인식 라이브러리	
주요 특징	빠른 속도와 쉬운 사용성	높은 정확도

개발	Google	CMU (카메기 멜론 대학)
모델 구조	Single neural network model	Multiple neural network models
속도	TensorFlow 기반으로, GPU 가속을 지원하여 상대적으로 빠름	높은 정확성을 위해 많은 파라미터, 메모리를 필요로 하므로 상대적으로 느림
정확성	33개의 주요 관절 추적 간단한 인체 추적에서 충분한 정확성	25개의 주요 관절 추적 복잡한 인체 구조 또한 정확히 추적

3.2 Unreal Engine

가장 널리 사용되는 게임 개발 엔진 중 하나로서, 높은 수준의 그래픽과 물리 시뮬레이션을 제공합니다. 게임 개발뿐만 아니라 가상현실, 영화 제작 등 다양한 분야에서 활용됩니다. *이 프로젝트에서는* Target 및 User의 움직임을 3D 환경에서 시각화하고 게임 형태로 UI를 구현하는 데에 사용되었습니다.

3.3 소켓 통신

소켓 통신은 네트워크 통신의 한 형태로, 서로 다른 프로세스 또는 머신 간에 데이터를 주고 받는 방법입니다. *이 프로젝트에서는* C++로 작성된 Unreal Engine과 Python으로 작성된 MediaPipe 간에 데이터를 주고 받는 데 사용됩니다.

이를 통해 Unreal Engine은 MediaPipe에서 인식한 사용자의 동작 데이터를 실시간으로 받아 게임 내에서 사용자의 움직임을 시각화하고 반응합니다. 반면, MediaPipe는 Unreal Engine에서 발생하는 게임 이벤트에 따라 사용자의 동작을 인식하고 분석하는 방식을 조정할 수 있습니다.

3.4 IK 및 FK

IK 및 FK는 물체의 움직임을 표현하기 위해 사용되는 기술입니다. *이 프로젝트에서는* Unreal Engine 기반 캐릭터에 IK를 기본적으로 사용하였으며, FK를 추가로 구현하여 각 랜드마크의 위치를 실시간으로 연산하여 게임 캐릭터에 적용하였습니다. 이를 통해 캐릭터의 움직임을 보다 자연스럽고 정확하게 표현할 수

있습니다.

IK(역운동학, Inverse Kinematics)는 주로 뼈대가 있는 오브젝트의 행동을 표현할 때 사용됩니다. 예를 들어 물건을 잡는 행동의 경우, IK는 물건을 잡는 손의 위치를 기준으로 Bone Tree의 루트까지 역으로 적절한 위치를 계산해 나갑니다. 이를 통해 캐릭터의 팔이나 다리 등을 목표 위치로 이동시킬 수 있습니다.

FK(정운동학, Forward Kinematics)는 IK와 반대로 루트로부터 목표 행동의 위치까지 적절한 관절의 각도를 계산하는 기법입니다. FK는 캐릭터의 뼈대 구조를 통해 각 관절의 각도를 계산하여 목표 위치에 도달할 수 있도록 합니다.

3.5 모션 인식

디지털 기기가 사용자의 움직임을 인식하고 이해하는 기능입니다. MediaPipe와 같은 도구를 사용하여 실현되며, 이 프로젝트에서는 사용자의 댄스 동작을 인식하는 데 사용됩니다.

4. 요구사항 분석

4.1 기능적 요구사항

4.1.1 모션 인식 기술

User의 움직임을 정확하게 감지하고 분석하기 위해 카메라 및 MediaPipe를 사용합니다.

- **정확성**: User의 움직임을 정확하게 인식 및 분석할 수 있는 성능 보장합니다.
- **실시간 처리**: 모션 인식과 분석 결과가 게임 플레이에 영향을 주지 않도록 최적화합니다.
- **다양한 환경 대응**: 조명, 각도, 배경, 의상 등 다양한 환경에서 안정적인 인식 보장합니다.
- **개인정보 보호**: 카메라를 활용하는 만큼 모션 데이터를 적절히 보호합니다.

4.1.2 실시간 점수 피드백

User의 움직임에 따른 실시간 점수 계산 및 피드백 제공 통해 게임성 및 사용자 경험을 향상시킵니다.

- **정확성**: 모션을 정확히 인식하고 그에 따른 공정한 계산 시스템을 필요합니다.
- **실시간 피드백**: 실시간 점수 및 피드백을 통해 즉각적으로 반응하고 유저 경험을 개선합니다.

4.1.3 사용자 친화적 UI

쉽고 직관적인 화면 구성을 통해 게임 플레이를 용이하게 합니다.

- **직관성**: 쉽게 이해 및 조작할 수 있도록 화면을 구성합니다.
- **디자인 요소**: 전체적인 분위기와 매력적인 비주얼을 마련합니다.
- **반응성**: 게임 속도 및 진행을 저하시키지 않는 빠른 반응을 보장합니다.
- **도움말**: 게임 및 인터페이스 이해에 도움을 주는 기능이 있는 것이 좋습니다.

4.2 비기능적 요구사항

- **성능**: 빠른 실시간 반응 및 애니메이션을 통해 원활한 진행을 도모합니다.
- **안정성**: 정상적으로 작동될 수 있도록 오류를 최소화하고, 예기치 않은 상황도 대응할 수 있도록 설계합니다.
- **사용자 친화적 설계**: 사용성 및 접근성 측면에서 사용자 요구에 맞도록 개선합니다.

5. 기능 구현

5.1 모션 인식 기능

본 프로젝트는 MediaPipe를 활용하여 사용자의 포즈를 인식하는 모션 인식 기능을 구현하였습니다. 카메라 모듈을 통해 사용자의 포즈를 캡쳐하고, 이를 3차원 벡터로 변환하여 언리얼 엔진에 맞는 좌표계로 변환하였습니다. 또한, 캐릭터 본에 따른 랜드마크를 적용하는 트리를 생성하여 부모와 자녀 간의 상대벡터를 계산하였습니다.

5.2 애니메이션 기능

- 캐릭터의 애니메이션 구현은 랜드마크 간 인접 로테이션을 표현하는 트리 구조를 활용하였습니다. 트리 구조는 골반, 가슴, 머리, 어깨, 팔꿈치, 손, 허벅지, 무릎, 발로 구성되었습니다. 트리의 루트는 골반으로, 골반과 가슴 노드는 선형 보간을 통해 추정하였습니다.

- 각 노드에 인접 노드 간 로테이션과 상대 벡터를 계산하여 저장하고, 키 값을 갖는 Map 구조에 트리를 저장하였습니다. 그 후, 계산된 로테이션과 상대 벡터를 애니메이션 블루프린트에 적용하여 각 뼈대에 로테이션을 직접 적용하였습니다.

해당 기능 구현 시 마주한 주요 어려움은 다음과 같습니다.

- Unreal Engine은 UObject 등 언리얼 클래스를 상속받아야 생성 가능하며, 일반적인 클래스를 허용하지 않습니다. 이로 인해 스마트 포인터, New/Delete 등의 가비지 컬렉션 기능이 제한되는 어려움을 겪었습니다.
- 랜드마크에서 골반과 가슴 등 필요한 랜드마크가 부재한 경우, 허벅지와 어깨 4개의 점을 이용하여 선형보간을 통해 골반을 추정하였습니다.
- 좌표축 문제로 캐릭터의 움직임이 부정확한 경우, 본의 위치에 따른 로컬 스페이스의 기저벡터 변화를 고려하여 좌표를 보정하는 작업을 수행하여 해결하였습니다.

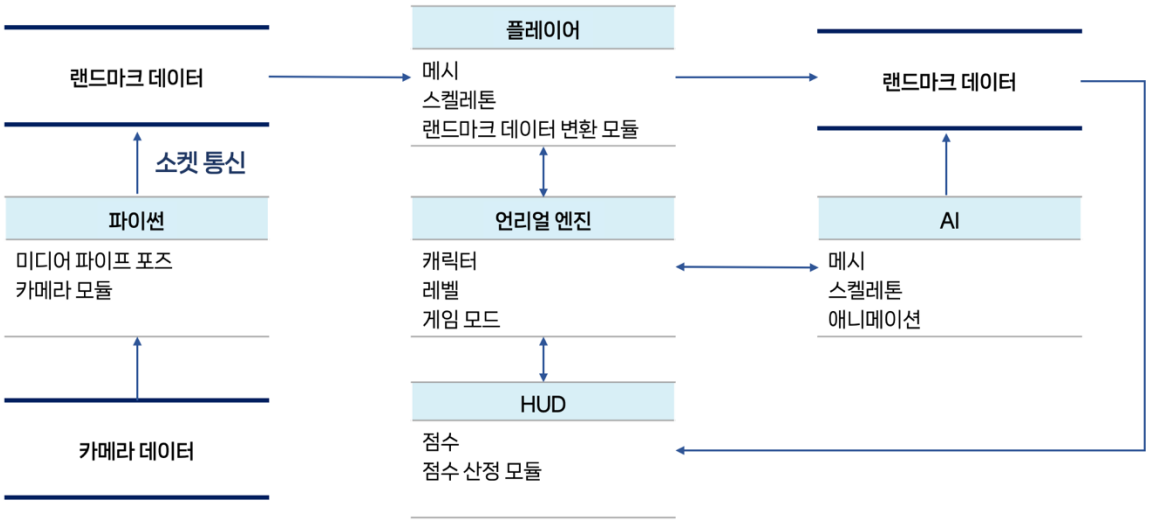
5.3 실시간 점수 피드백 기능

Target 포즈 및 User 포즈 Mesh의 relative location을 비교합니다. Target 포즈와 User 포즈의 상대적인 뼈 위치를 비교합니다. 이 때 비교하는 뼈의 위치는 neck, 양쪽의 lower arm, middle, hand, calf, foot입니다. 자세한 기능 구현은 다음과 같습니다.

- GetAllSocketName() 함수를 통해 양 쪽의 모든 뼈 정보를 가져옵니다.
- CompareValues() 함수를 통해 Target 포즈와 User 포즈에서의 값을 비교합니다.
- 값을 비교했을 때 오차 값이 허용되는 범위에 포함되어 있다면 true, 허용되는 범위 밖에 있다면 false를 return합니다.
- true 또는 false 값을 매 초 반환하며, 스테이지 플레이 시간이 N초일 때, true 및 false는 스테이지 종료 시 합쳐서 N회 반환됩니다.
- true가 반환된 횟수, false가 반환된 횟수의 비율에 따라 최종적으로 A, B, C의 등급을 부여 받습니다.

6. 시스템 아키텍처

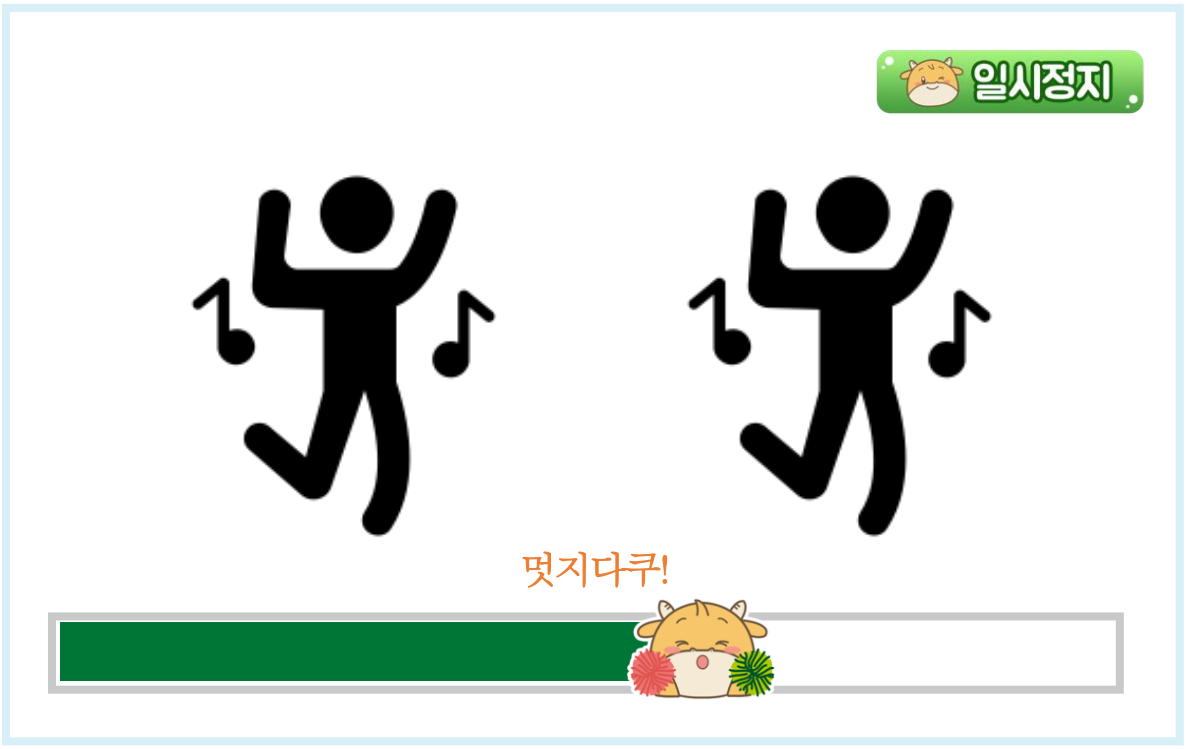
본 프로젝트의 전체적인 구조를 나타내는 시스템 아키텍처는 다음과 같습니다.



7. 화면 디자인

7.1 타이틀 화면 디자인

7.2 스테이지 화면 디자인



7.3 결과 화면 디자인

8. 한계 및 성취

8.1 한계

MediaPipe의 포즈 인식 기능은 다양한 시나리오와 환경에서의 실용성을 증명하기에 앞서, 여전히 몇 가지 도전 과제와 한계점을 갖고 있습니다. 특히, 인식 알고리즘이 사람의 전면과 후면을 혼동하며, 이로 인해 가끔 비정상적인 랜드마크를 생성하는 현상이 발견되었습니다. 이러한 혼동은 캐릭터의 동작과 포즈에 부자연스러운 떨림과 왜곡을 유발하였습니다.

또한, 애니메이션 구현 과정에서 Forward Kinematics를 골반을 중심으로 구현하는 상대벡터 방식이 적용되었지만, 이는 캐릭터가 뒤로 도는 모습을 효과적으로 구현하는 데 한계를 드러냈습니다. 이는 랜드마크로 구성된 상대 벡터의 트리 구조만으로는 캐릭터의 전체적인 방향성과 공간적 위치를 정확히 인지하는 데 어려움이 있었음을 보여줍니다.

특히, 이런 문제는 캐릭터가 숙이거나 높이를 변화시키는 등의 복잡한 동작을 표현할 때 더욱 도드라져, 캐릭터가 마치 공중에 떠 있는 것처럼 보이는 시각적 문제를 초래하였습니다.

8.2 성취

미디어 파이프의 포즈가 2D 이미지에서 좌표를 생성하는 데는 뛰어나지만, 그것을 3D 환경에 적용하는 것은 복잡하고 어려운 과제였습니다. 하지만, 이런 도전 속에서도 우리는 랜드마크를 정규화하고, 뼈대에 대한 트리 구조를 만들어내는 데 성공하였습니다.

트리 구조를 바탕으로 인접 노드 간의 상대 벡터를 생성하고, 로테이션을 계산하며, 각각의 뼈대 구조에 맞게 조정할 수 있었습니다. 이에 다양한 카메라 환경에서도 적절한 포즈를 모사하는 데 성공할 수 있었습니다.

또한 이것을 바탕으로 Target 포즈와 User 포즈 간 유사도를 계산하고 실시간 점수 피드백 및 최종 등급 산출이 가능하도록 구현할 수 있었습니다. 다음은 각 포즈 간 유사도가 높을 때와 낮을 때 각각의 CASE에 대한 실제 플레이 화면입니다.

[CASE : 동작 유사도가 높은 경우]

[CASE : 동작 유사도가 낮은 경우]

이상입니다.