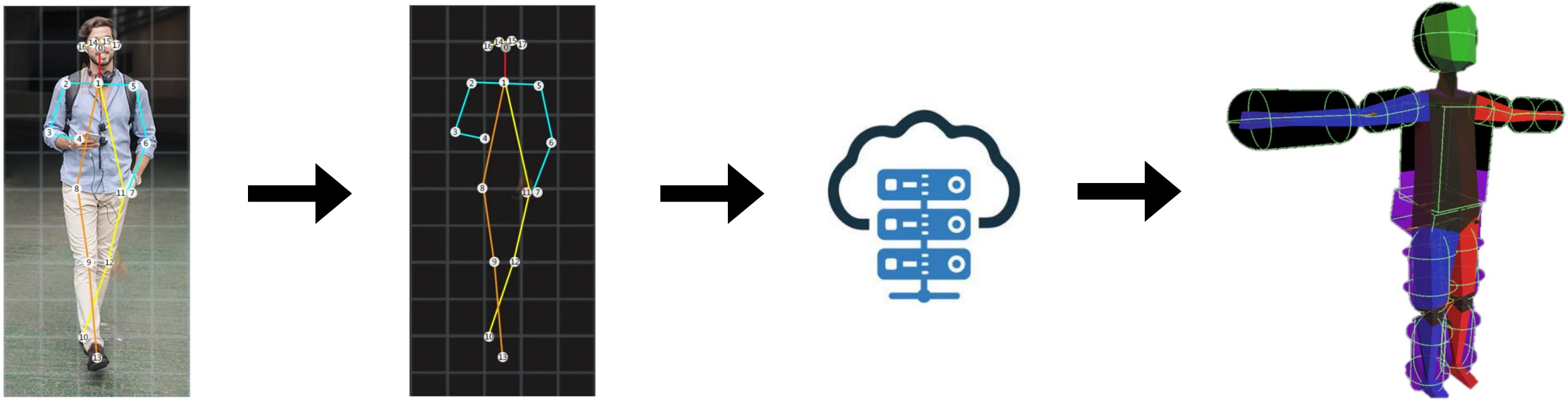
미니 게임

3조 박규남, 유임성, 황인경

Part 0

프로젝트 개요

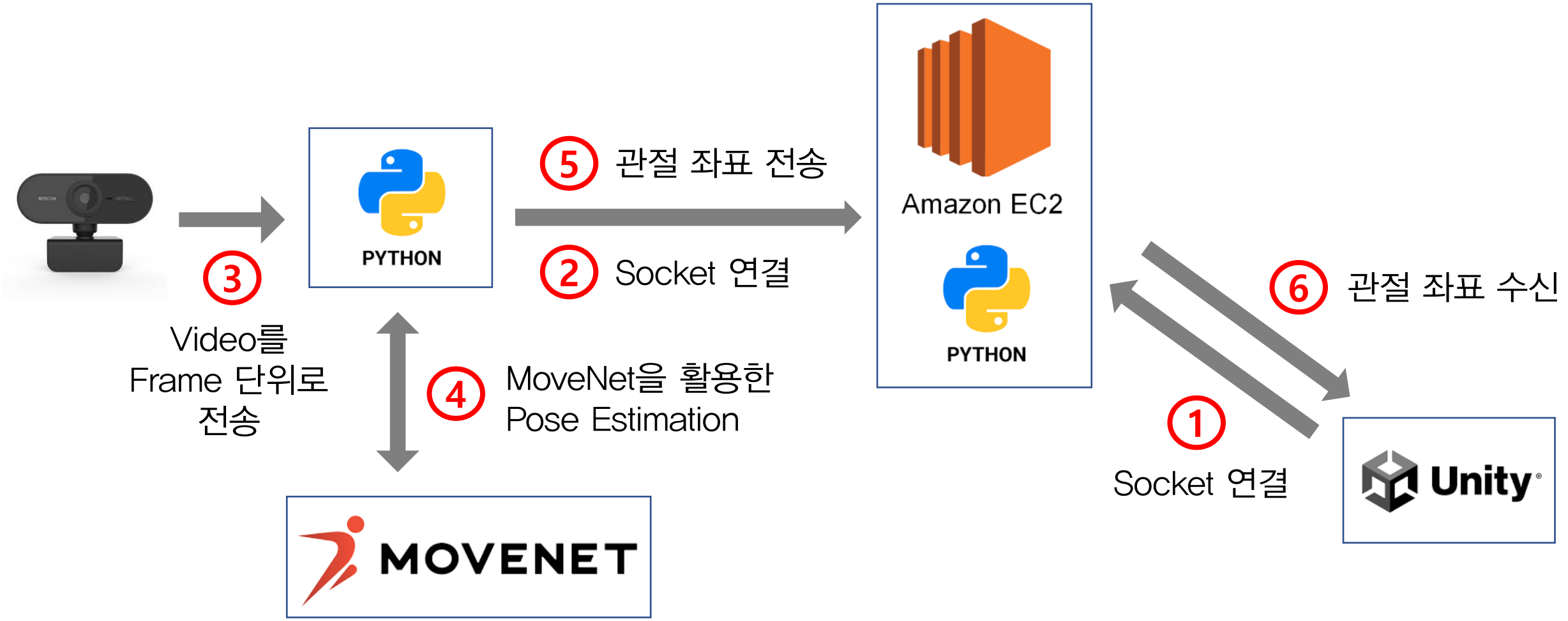
- Deep Learning을 통한 Real-time Human Pose Estimation의 구현
- Socket 통신을 활용한 Python → Unity 데이터 전송
- Pose Estimation 데이터를 활용한 미니게임 구현



목차

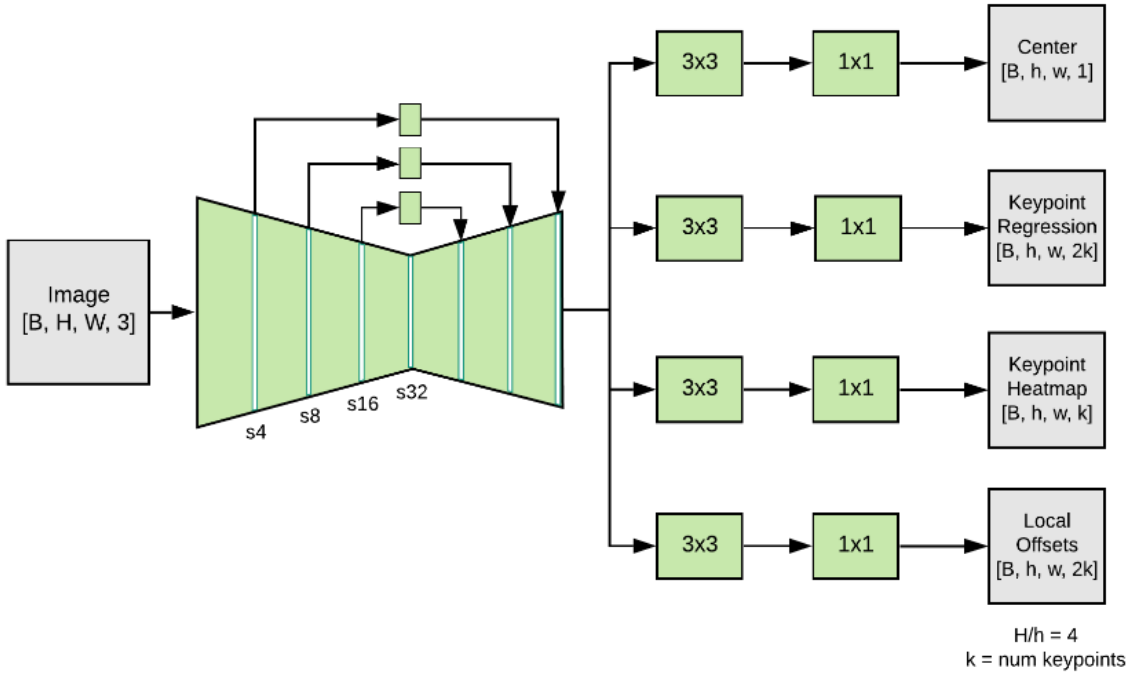
Part 1

Pipeline



Part 2

MoveNet & Realtime Pose Estimation



- **Architecture**

  Feature extraction + Prediction

  (MobileNet2+FPN)

- **Prediction**

  4–step Prediction

  Top–down Method

- **Realtime Pose Estimation**

  Camera : Webcam

  Speed : Approx. 30 FPS

  Data : Joint Keypoints with scores

Part 3

Unity

- **Character**

  Skin Mesh + Capsule(Collider)

  Capsule : 게임 내 충돌 판정을 위하여 필요

- **Character Animation**

  MoveNet(딥러닝)을 통해 얻은 좌표를 활용

  캐릭터 및 Collider를 구성하는 각 오브젝터의

  회전은 두 관절의 좌표를 연결하는 벡터와

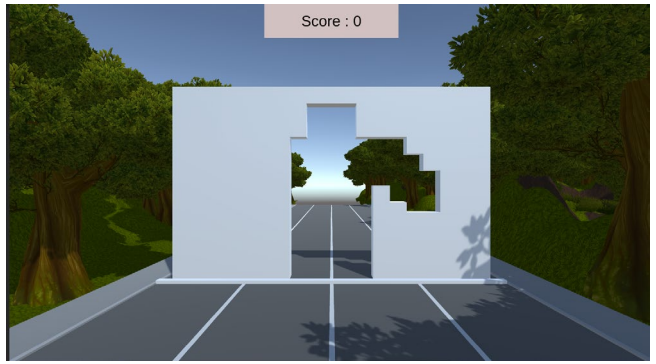
  X축이 이루는 각도를 활용해 설정 가능

Part 3

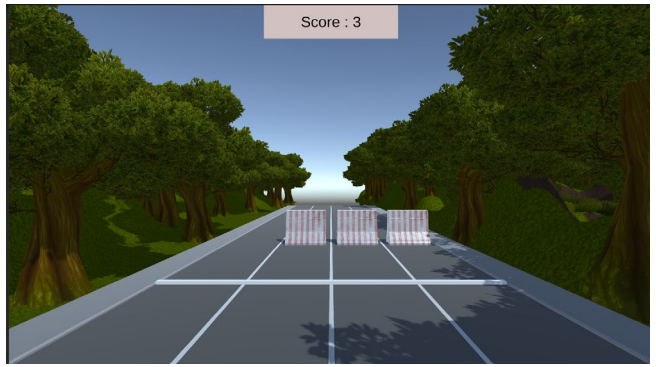
Unity

메인 화면

두뇌 풀가동



알투비트



물건 피하기

게임 오버

Part 4

서버 구축

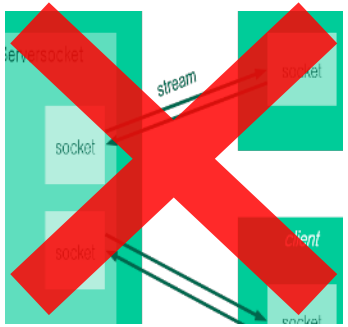
- EC2에 MoveNet(Python)과 유니티의 Frame 데이터 통신을 매개할 Python 서버 생성
- 유니티와 Python의 Socket 연결을 Listen
- 유니티와 Python의 Socket 연결이 들어오면 해당 연결을 유지
- MoveNet이 Frame 데이터를 생산하기 시작하면 Python에서 해당 데이터를 서버로 전송
- 서버는 전달받은 데이터를 가공 없이 그대로 유니티로 전송
- 유니티 클라이언트는 연결된 Socket의 NetworkStream에서 Byte 데이터를 받아옴
- 받아온 데이터를 적절히 가공 및 수정하고, 1프레임에 해당하는 데이터를 사용하여 캐릭터의 자세를 알맞게 수정

Part 5

힘들었던 점

- GPU 사용의 제한
- GPU의 여부에 따른 Inference Time의 차이가 큼
- OpenPose는 GPU 사용 없이 충분한 FPS를 달성하지 못함

➢ CPU를 활용해도 충분한 속도를 보이는 MoveNet 모델 사용



- Socket 통신에서 데이터를 송수신할 때 Stream을 사용하였으므로 정확히 한 프레임에 해당하는 데이터를 가져오기 어려움

➢ 프레임 데이터 사이에 구분자(! 문자)를 추가하여 문제 해결

- 정확성이 높은 모델(OpenPose)을 사용할 때 Inference Time을 향상시키기 위해 클라우드 GPU를 제공하는 구글 코랩을 활용함

➢ GPU를 사용하였음에도 여전히 처리 속도가 부진하여 최종 모델을 MoveNet으로 결정

Part 6

게임 시연

Appx.

프로젝트 코드

- 유니티 클라이언트: https://github.com/imsongkk/APF-UnityClient
- EC2 파이썬 서버: https://github.com/imsongkk/APF-Server
- Webcam Pose Detection: https://github.com/imsongkk/APF-PoseDetection

감사합니다.