Algorithm 1: Preprocessing of skeleton animation data

Input:

skeleton;

animations;

Output:

BonesAffineMatrixs ;

```
1 for frame = animations[animType][frameIndex] do
2     Update skeleton by frame;
3     for bone = skeleton[boneIndex] do
4         matrix = bone.affineMatrix
5         while bone have parent do
6             bone = bone.parentBone;
7             matrix = matrix × bone.affineMatrix
8         BonesAffineMatrixs[animType][frameIndex][boneIndex] = matrix;
```

Algorithm 2: Realization of diversity of crowd animation

Input:

modelGeometry;

crowdParameter;

BonesAffineMatrixs ;

animationPlayTime ;

Output:

crowdGeometry ;

```
for avatarParameter ∈ crowdParameter do
    GET affineMatrix, animationType, animationSpeed
    FROM avatarParameter.
    for vertexInf ∈ modelGeometry do
        GET position, boneIndex, coordinateUV
        FROM vertexInf.
        if bones[boneIndex] have animation then
            numberOfPlayedFrames=rounding( animationPlayTime*animationSpeed );
            frameIndex = numberOfFramesPlayed mod frameIndexMax;
        else
            frameIndex = 0;
        animationMatrix = BonesAffineMatrixs[animationType][frameIndex][boneIndex];
        vextexScenePosition = avatarMatrix × animationMatrix × position;
        crowdGeometry.push(vextexScenePosition);
```

Algorithm 3: Partition binding of avatar texture map

Input:

modelGeometry;

crowdParameter;

textureMapping;

Output:

crowdMaterial ;

```
for avatarParameter ∈ crowdParameter do
    GET position, coordinateUV
    FROM avatarParameter.
    for vertexInf ∈ modelGeometry do
        GET headTextureType, upperBodyTextureType, trousersTextureTypee , neckHeight , waistHeight
        FROM vertexInf.
        if position < waistHeight then
            textureType = headTextureType;
        else if position < neckHeight then
            textureType = upperBodyTextureType;
        else
            textureType = trousersBodyTextureType;
        crowdMaterial.push(textureMapping[textureType][coordinateUV]);
```