Algorithm 1: Preprocessing of skeleton animation data

Input: skeleton, animations;

Output: BonesAffineMatrixs ;

```
for frame ← animations[animType][frameIndex] do
    Update skeleton by frame;
    for bone ← skeleton[boneIndex] do
        matrix ← bone.affineMatrix
        while bone have parent do
            bone ← bone.parentBone;
            matrix ← matrix × bone.affineMatrix
        end
        BonesAffineMatrixs[animType][frameIndex][boneIndex] ← matrix;
    end
end
```

Algorithm 2: Diversifying 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;
        end
        animationMatrix ← BonesAffineMatrixs[animationType][frameIndex][boneIndex];
        vextexScenePosition ← avatarMatrix × animationMatrix × position;
        crowdGeometry.add(vextexScenePosition);
    end
end
```

Algorithm 3: Partitioning avatar texture map

Input: modelGeometry, crowdParameter, textureMapping;

Output: crowdMaterial ;

---

for $avatarParameter \in crowdParameter$ **do**
  GET { $textureType_{\{head,upperBody,trousers\}}$, $height_{\{neck,waist\}}$} FROM avatarParameter.
  for $vertexInf \in modelGeometry$ **do**
    GET {position, coordinateUV } FROM vertexInf.
    if $position < height_{waist}$ **then**
      $type \leftarrow textureType_{head}$;
    else if $position < height_{neck}$ **then**
      $type \leftarrow textureType_{upperBody}$;
    else
      $type \leftarrow textureType_{trousers}$;
    end
    $crowdMaterial.add(textureMapping[type][coordinateUV])$;
  end
end