CPSC 314
Assignment 2, Part 2 : Magical Deformations

Due 11:59PM, February 9th, 2020

1 Introduction

A dangerous armadillo has arrived in the scene to scare the wizard! In this assignment, you will be deforming different parts of the armadillo using your knowledge of transformations.

1.1 Getting the Code

Assignment code is hosted on the UBC Students Github. To retrieve it onto your local machine navigate to the folder on your machine where you intend to keep your assignment code, and run the following command from the terminal or command line:

```
git clone https://github.students.cs.ubc.ca/cpsc314-2019w-t2/a2-part2-release.git
```

1.2 Template

- The file `A2-P2.html` is the launcher of the assignment. Open it in your preferred browser to run the assigment, to get started.
- The file `A2-P2.js` contains the JavaScript code used to set up the scene and the rendering environment. You will need to make small changes in it to answer the questions.
- The folder `glsl` contains the vertex and fragment shaders for the wizard, eyes, scorch mark geometry, and magic circle geometry. New shaders have been added for deformation, called aramadilloNeckDeformShader and armadilloSkinningShader.
- The folder `js` contains the required JavaScript libraries. You do not need to change anything here.
- The folder `obj` contains the geometric models loaded in the scene.
- The folder `images` contains the texture images used.

- The folder `armature` contains a file of type **gltf**. The gltf file contains the armadillo geometry as well as skeleton we will use to deform the armadillo. All this is loaded into the scene, a keypress '2' brings the armadillo into the scene.

2 Work to be done (60 points)

First ensure that you can run the template code in your browser. If you have any issues, see instructions in Assignment 1. This assignment has two parts, the two parts are independent of each other.

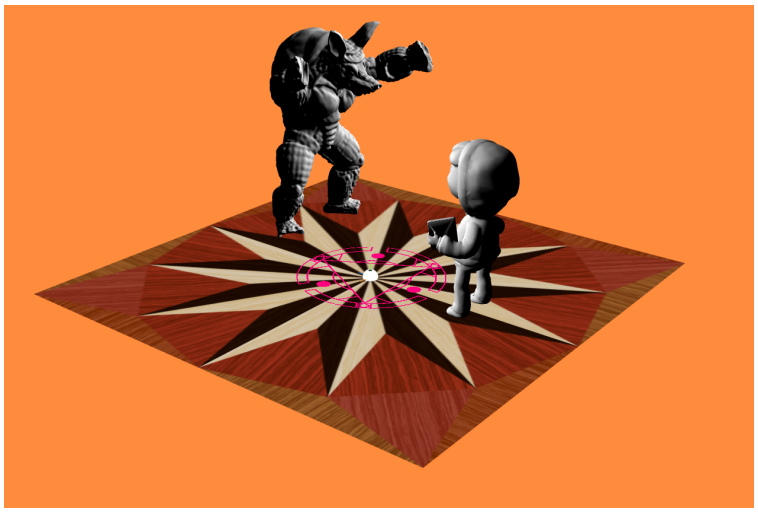


Figure 1: The initial scene of question 1 and 2

Part 1: (30 points) Required Features

1. **(30 points)** The threatening head nod
   In this question, you task is to simply rotate the armadillo's head about a line through the neck, parallel to the x-axis. You will have two coordinate frames, the neck and head frame provided to you. The neck frame is fixed to the body, while the head frame is fixed to the head and moves with it. The armadillo doesn't actually have a neck, an approximation of the origin such as (0.0, 6.5, 1.28) is sufficient and the X, Y, Z the axis of the neck coordinate frame are aligned with the world frame.
   You will have to first determine if a specific vertex is part of the head or the neck. One way is to find an axis aligned bounding box for the neck. To find an axis aligned bounding box, find a range for each dimension(X, Y, Z) such that the vertices of the head lie within the range.
   The `noddingArmadillo.vs.glsl` shader already gets you started on the code for finding the bounding box of the armadillo's head. Study it and make sure to understand it before proceeding to the next part.

(a) Bounding box for the head

(b) Deformation of the neck

Figure 2: Question 1

Once you've identified the vertices that belong to the head frame, your **main task** is to rotate the head frame relative to the neck.

Hint 1: Create a rotation matrix for the head frame.

The angle at which the Head frame is rotated relative to the Neck should be controllable by the keyboard. Make sure to add a new variable and correctly pass it to the shaders. Don't panic if some significant artifacts pop up around the boundary between the head and neck for larger rotations.
You should pass the correct uniforms to the `noddingArmadilloMaterial` and modify `noddingArmadillo.vs.glsl` for this task.

2. **(30 points)** Linear Blend Skinning

Linear blend skinning is one of the most common algorithms used to deform meshes which have an underlying skeleton. This is used to make 3D characters in video games and other real time applications. *We will go over this technique in class.*

In this question you will get to try out linear blend skinning on the armadillo.

- You are given the skeleton for the armadillo, you are also given the the influence of each bone on each vertex. In the `skinningArmadillo.vs.glsl` shader, these are called weightBone1 and weightBone2.

  Hint : Notice that this is procedurally calculated in the vertex shader. For example, all vertices below the armadillo's hip are assigned a weight of 1 for the influence of the hip bone.

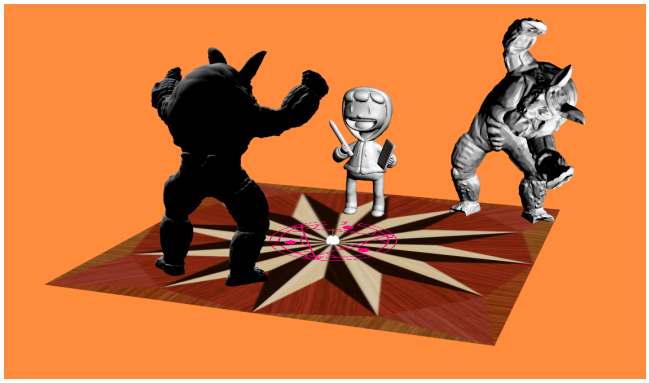
- You are given the world transforms (stored in `boneMatrices`) of both bones and the inverse of their respective bind matrices (stored in `inverseBindMatrices`).

Note *: Keypress '1' brings the new armadillo into the scene.*

The armadillo has only 2 bones, as a result every vertex has 2 transforms to blend, and 2 weights to decide how to blend them. Bone1 is the parent of Bone2. (In real world applications, the armatures get very complex.)

The essential part of this skinning algorithm is to build the transformation matrices for each bone of the skeleton and pass the transformation matrix as a uniform to the skinningShader.

For this question, you will build a transformation matrix for each bone and apply a weighted combination of these transformations in the vertex shader. You will be modifying the `setMatrices` function and the `armadilloSkinning` vertex shader.



(a) Skinning applied to the armadillo

Figure 3: Question 2

(a) Your first task is to complete the `armadilloSkinning.vs.glsl` shader. The transformation matrices for the two bones are passed as uniforms to the shader. The weights per bone are stored in the variables `weightBone1` and `weightBone2`. Find the weighted combination of the transformation matrices from the uniform array and apply it to the vertex position. Make sure to multiply the final position by a global Translation matrix to correctly position the armadillo in the scene.

Hint 1: A uniform to store the transforms of the bones is already created and passed to the shader
Hint 2: To get the influence of the first bone on the vertex, multiply its transformation by `weightBone1`

Once this step is done, a keypress of "M" and "N" will rotate the lower body of the armadillo with respect to the X axis. This is because the transformation

of the first bone is getting updated and passed to the shader in the `setMatrices` function.

(b) In this question, you will update the transformation matrix of the second bone and rotate the armadillo's upper body. Carefully study the construction of the transform1 matrix in the setMatrices function. The transform1 matrix first copies the world transform of bone1, this is stored in `boneMatrices[0]`. Some custom transformations are applied to the world transform (rotation about X) the inverse of the bind matrix is then multiplied. This matrix is then sent to the shader as a uniform.

Your task is to build the transformation matrix for the second bone. It follows the same steps as transform1, but with an additional step of copying the parent matrix. Recall that bone1 is the parent of bone2. Using transform2, add a rotation about X and Z axis to the armadillo's upper body. This rotation should be updated with a keypress.

Hint 1: First step to building the transform for bone2 is to copy or multiply its parent's matrix
Hint 2: After the first step, remember to multiply bone2's world matrix, stored in `boneMatrices[1]` *before adding any other custom transformations.*
Hint 3 : The correct values for the bone matrices and the inverse bind matrices are provided for you.
Hint 4 : If you're curious about what these bone matrices and their inverses are, check the gltf loader at the top of the js file.

The main advantage of linear blend skinning is that it is not computationally intensive and it still provides good enough deformations, but as you may have noticed its not without faults.

2.1 Hand-in Instructions

You will be handing in both parts of A2 together. You do not have to hand in any printed code. Create a README.txt file that includes your name, student number, and CWL username, and any information you would like to pass on to the marker. Create a folder called "A2" under your "cs-314" directory. Within this directory have three subdirectories named "part1,""part2", and "creative" and put all the source files, your makefile, and your README.txt file for each part in the respective folder. The assignment should run without any changes directly from your submission folders. The assignment can be handed in on a department computer, which you can SSH into, with the exact command:
`handin cs-314 A2`

You may also use Web-Handin by following this link
`https://my.cs.ubc.ca/docs/hand-in,`

logging in with your CWL credentials, and writing cs-314 for the course, A2 for the assignment name, and zipping your assignment folder for submission. It is always in your best interest to make sure your assignment was successfully handed in. To do this, you may either use the Check submissions button in Web-Handin, or using the -c flag on the command line `handin -c cs-314 A2`.