UNIVERSITÀ
DI TRENTO

Department of Information Engineering and Computer Science

Project Course on Computer Graphics

3D HUMAN SKELETON COMPOSED OF JSON'S

Tosato Lucrezia 220370

Academic Year 2020/2021

Contents

Chapter 1

Introduction

In this project, I build a skeleton composed of a pre-defined structure of:

- 17 joints
- 16 bones' connecting joints

The final goal of the project was to create a program capable of reading at least one JSON file and show the figure of a skeleton, a JSON file is a textual format for structuring data that allow to master information in any area of Computer Science.
This model includes an environment consisting of stylized figures of the ground, a hat, a chair, a table, a glass and a plate.
The program reads a series of JSON files sequentially, creating the motion effect. Once it gets to the last file, it resets and restarts from the first one.
It is possible to move the skeleton and the environment on all axes except the negative y-axis, creating the effect of a real floor.
An overview of the model within the full environment is displayed in Fig1.1.
Cap.2 introduces the structure of the program ans some key features are discussed in detail in Cap.3.

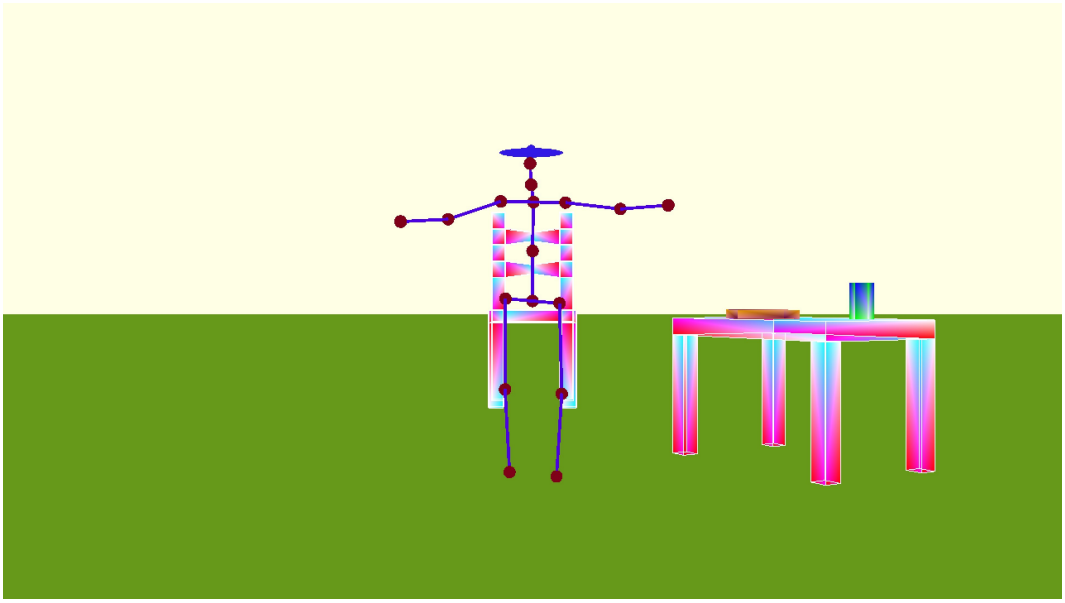


Figure 1.1: Skeleton in the enviroment

1.1 Packages to install

First of all it is necessary to have Python installed on the device, at this point it is required to install:

- PyOpenGL
- numpy
- pygame

Once you have downloaded these packages you only need to have all the files in the same place together with the folder called "animation" and compile the program "main.py".

Chapter 2

Program Structure

The program is implement in Python version 3.9.1 on Atom and PyCharm using PyOpenGL library. For the development of the program I referred to the tutorials offered by Sentdex[1], Auctux[2], AtiByte[3] and the material made available by the professor.

2.1 Diagram

An intuitive diagram on how the functions are organized is represented in the Fig.2.1.

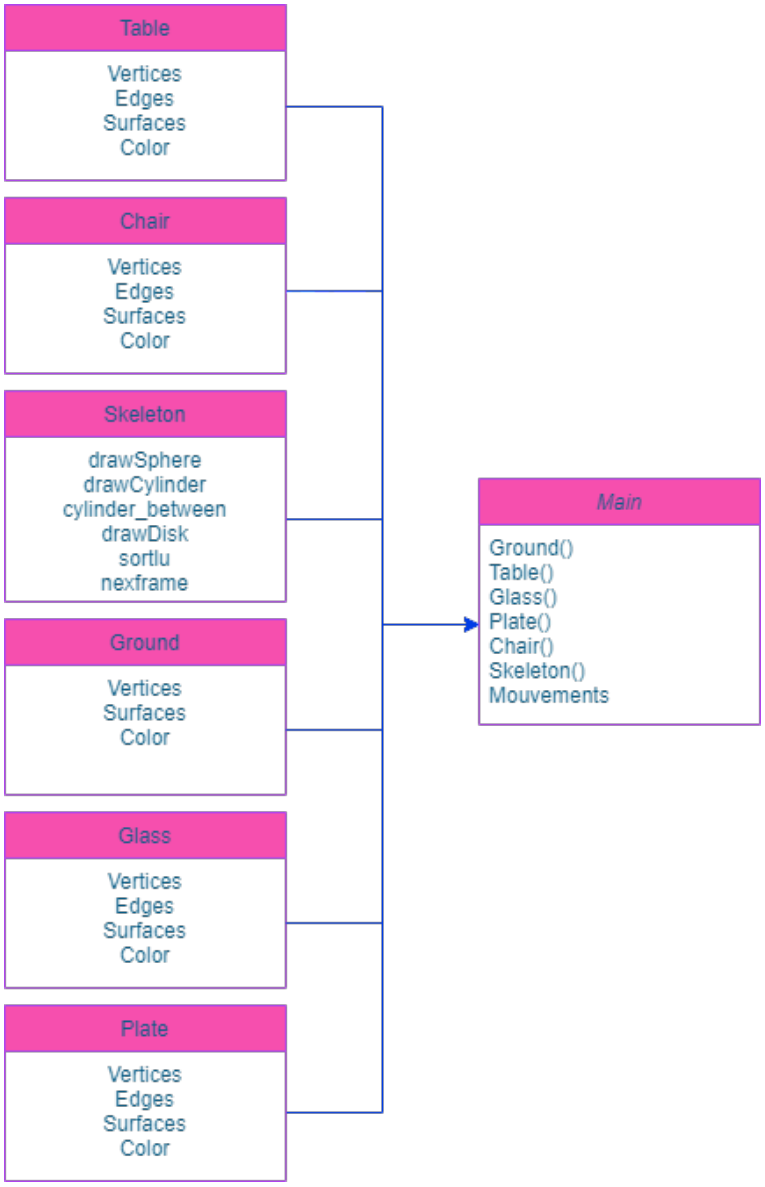


Figure 2.1: Diagram

Chapter 3

Key Features

3.1 Skeleton

For the creation of the skeleton the program starts by reading for each JSON file the coordinates of all the points, then it creates for each of these a sphere that is then translated into the correct position. To design the cylinders I started by drawing one for each sphere perpendicular to the ground, to calculate the proper length of each one I used the formulas:

$$p(A, B) = [(x_A - x_B), (y_A - y_B), (z_A - z_B)]$$

$$h(A, B) = \sqrt{(p[0])^2 + p[1])^2 + (p[2])^2}$$

with **'A'** and **'B'** the first and second vertices, respectively, **'p'** a vector of 3 elements and **'h'** a value. Which can be simplified into the well-known formula:

$$d(A, B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2 + (y_A - y_B)^2}$$

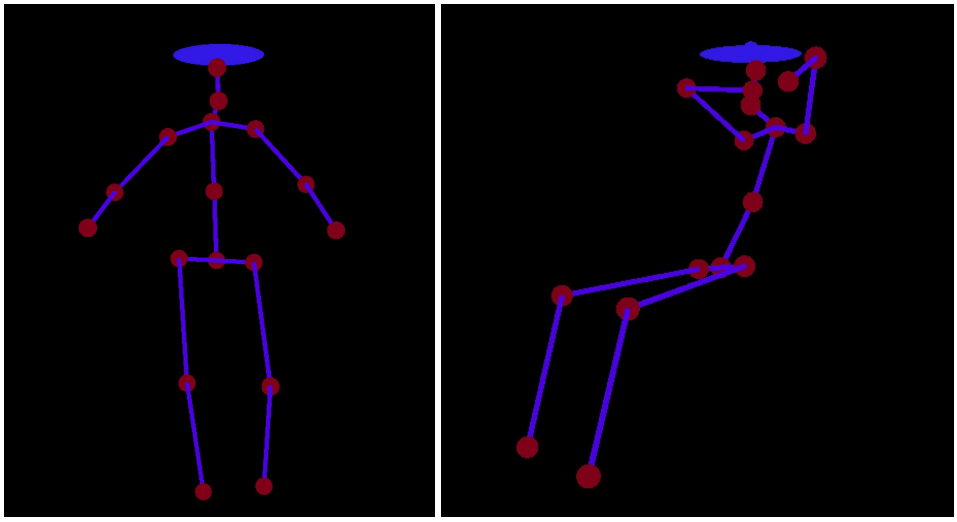
Then, to give the right inclination to each cylinder I implemented a function that takes as input the start point and the end point of the cylinder, calculates the hypotenuse and checks if it is greater or less than 0.01. For smaller values the script consider the direction of the cylinder equal to the default direction on PyOpenGL, otherwise, it calculates it through the cross-correlation between the point-to-point distance vector and the default direction. The angle is calculated as:

$$angle = -\arctan\left(hypot(p[0], p[1]), p[2]\right)$$

Once all these parameters are calculated, everything is saved in lists that are given as input to the functions to draw spheres and cylinders. At this point within the skeleton function, after having reordered the JSON files with a sort function that uses a specific key parameter, all the files are read, creating the effect of movement.

3.1.1 Hat

A stylized hat was added to dress up the skeleton, created by joining a disk and a sphere. The object is placed only in the coordinates of the 10th point, which represents the head, raised by a certain height. Inside the function for each pair of points are calculated the parameters previously illustrated, creating as final result the skeleton in Fig.3.1 composed of 17 spheres, 17 cylinders (the cylinder starting from the tenth sphere has zero height) and a hat.

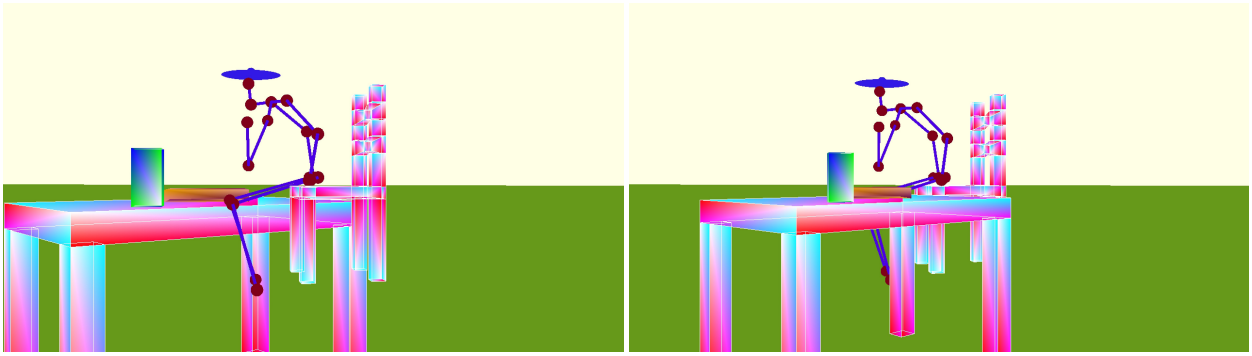


(a) (b)

Figure 3.1: Skeleton

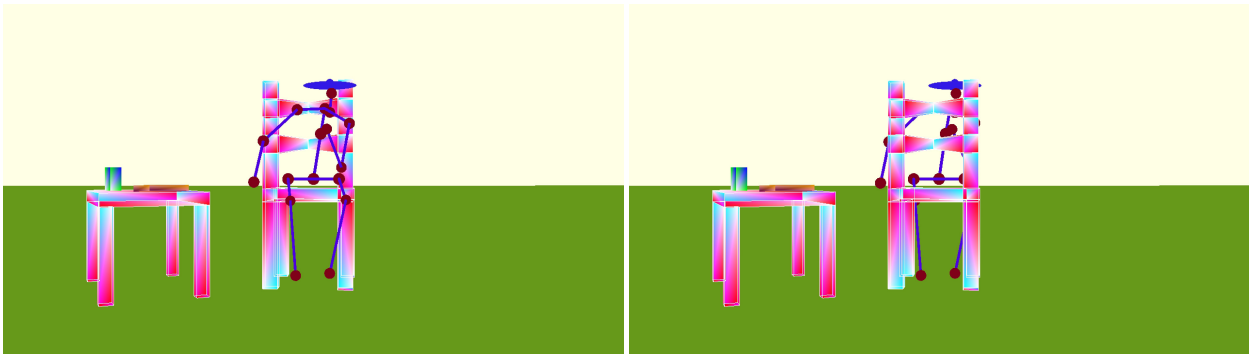
3.2 Enviroment

Controllers have been put in place that prioritize one function over another based on the angle at which the camera is facing the scene. In this way an object is not transparent compared to another, and the depth of the scene is always maintained as shown in Fig. 3.2 where the skeleton is sitting behind the table relative to the camera perspective, and in Fig.3.3 where the camera has its back to the skeleton so the chair has to cover the skeleton.



(a) Before (b) After

Figure 3.2

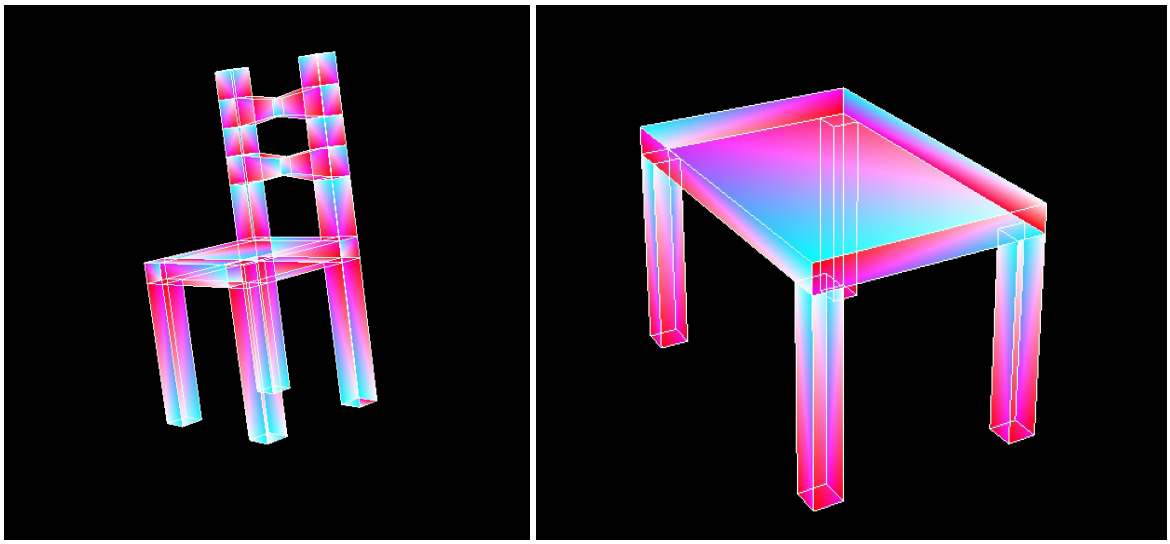


(a) Before (b) After

Figure 3.3

3.2.1 Chair and Table

The chair was designed thanks to the tutorial proposed by Auctux[2], using Blender and exporting the vertex values to python with a simple script.
The biggest difficulty of this process is to be able to identify the correct measurements of the object that will then be exported. For this reason the table was eventually designed without using Blender, it is possible to set the measurements of the table within its file without having to manually edit each vertex.

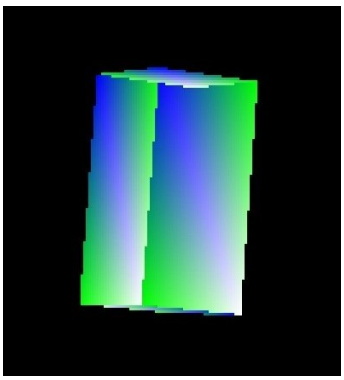


(a) Chair (b) Table

3.2.2 Ground, Glass and Plate

For the construction of the ground, the glass and the plate it was enough to create rectangles with different sizes and translate them to the desired positions.
Initially the plate and the glass were drawn on Blender but the large number of vertices created by drawing cylindrical shapes created a lot of problems, so I opted for the simplest solution as shown in Fig.3.5.
As with the table, it is possible to set their measurements directly by changing a single variable without having to modify all the vertices.

(a) Dish



(b) Glass

Figure 3.5

3.3 Music

Code lines 139-140, which allow the insertion of background music to the scene, have been commented out.
The choice was made for two reasons:

- On GitHub it is not possible to share music tracks due to copyright, the lack of the track would have created problems in the compilation.
- The second reason is that while on Ubuntu there are no problems in the compilation, on Windows there can be problems caused by the IDE.

It is however relevant to keep in mind that it is possible to use this feature by inserting the song in the same folder of the code and uncommenting the two lines.

3.4 Camera Focus and Movement

Once the program is started the skeleton and the environment advance up to a certain distance from the camera, once arrived at this distance the skeleton starts to move and it is possible to start changing the focus of the camera moving the model, the commands are:

- **'mouse wheel forward'**: the model moves along the positive x-axis, zoom in effect
- **'mouse wheel back'**: the model moves along the negative x-axis, zoom out effect
- **'w'**: rotate the modele down
- **'z'**: rotate the model up
- **'s'**: rotate the model to the left
- **'a'**: rotate the model to the right
- **'right arrow'**: move the model to the left
- **'left arrow'**: move the model to the right
- **'up arrow'**: move the model down
- **'down arrow'**: move the model up

All controls were chosen to give the impression of moving a camera instead of the scene.

Chapter 4

Conclusions

The project performs the required functionality but it is certainly not comparable to work done with other languages, in particular with regard to the jerkiness of the camera movements.
Using python allows to write a code more easily understandable and simple, as well as very easy to compile, but being an interpreted language has less power and speed.
In general I am satisfied with the work done, in particular the preservation of the depth of the environment and the objects placed around the skeleton, finally, considering that most of the material on OpenGL refers to the C++ community finding usable information was not easy and there were several failed attempts to add a SkyBox to the scene.
In conclusion I would say that the program works, it could be improved perhaps inserting the model inside a SkyBox or designing more realistic objects to be placed around and on the skeleton. Probably using a desktop computer, with more computing power, it would have been possible to do this through Blender by exporting the vertices of the objects.

4.1 Demo Video

A demo of how the project works can be found in a video I made: *Youtube Video*.
For screen recording I used *Streamlabs*.

4.2 GitHub Repository

All the code is also accessible on github: *Computer_Graphics_Skeleton_Enviroment Repository*.

Bibliography

[1] SENTDEX. OPENGL WITH PYOPENGL TUTORIAL PYTHON AND PYGAME
Youtube Video
Website

[2] AUCTUX. MAKE 3D GRAPHIC WITH PYTHON OPENGL
Youtube Video
GitHub Repository

[3] ATIBYTE. OPENGL IN PYTHON
Youtube Video
GitHub Repository