Vishwakarma Institute of Technology, Pune.

Department of Engineering, Sciences and Humanities (DESH)

R3ND3R : Molding, Handling and Accessing Multi-Dimensional Models

Authors : Mst. Shreyash Bhatkar (Mail : shreyash.bhatkar19@vit.edu) and
Ms. Shreya Hiremath. (Mail : shreya.hiremath19@vit.edu)

Second Year, B.Tech. VIT, Pune. EnTC Engineering.

Abstract *—*Graphics are one of the most popular sources of data transfer nowadays, we can arguably term them as the easiest data to read, and herein we are dealing with the same, in our project R3ND3R, we are focusing more on making an impact and it is obviously going to set an indelible benchmark in learning, teaching, developing, designing and automation. What's really cool about it is how easy the application makes it to control everything using Python. The use of 3D modelling software such as Blender has increased over time for games, simulations and animations. Such software allows graphical models to be developed with ease and to be visualized immediately rather than at run time.

The project which we have worked on aims to create an interface for visualizing any sort of three dimensional or two dimensional data for a better understanding of the models which we create and their physical attributes. Thus there is no holds barred in any way, if we are good with python programming we can sort of create any kind of shape we want, also we have focused more on providing an interface for the user and handling our data from one place for the convenience measures.

Keywords *— Animation, Face Mocap, 3-D models, Open CV, Python, Blender, Supershapes, Mesh, Tkinter, etc.*

I. INTRODUCTION

We are in the era of computation and simulations, the need of the hour is to create data designs which are simple to read for the user and to create for the manufacturer as well, suppose we want an example of this, let us consider a factory project at BMW, we are being suggested by a prototype by the research and development team and we actually think it's a good idea but you have certain doubts you need to get over before implementing the actual design so in terms of physicalities of that stereotype, so you create a three dimensional model and then analyse it rather than using actual hardware so that you can avoid risks of failure and also save some capital.

Regularly 3D simulation of various models and data analysis offers more facilities to the students, scientists, graphic designer etc., when contrasted with 2D. while observing the output/ results in form of 3D gives a more realistic view with textures and lights, and can be through multiple angles. Therefore , its more convenient to create 3D simulations.

We used blender 2.8 in this project and there were several other modes like unity which we considered but for the ease .Thus to compare between both the modes we have tried here, as we have tried both blender and unity, we observed that both of these softwares do alter in their application, and on a personal level, we found scripting in blender quite easier as compared to unity. Also unity dealt with basic language support, i.e. C# , JavaScript , etc. Same goes with blender, still it supports python libraries across scripting scenario, thus as we find python easier here, we chose to work with the blender API. And most importantly we can use python libraries into it while creating models so we were comfortable with python so we chose blender and interface was user friendly as well.

And looking from a consumer point of view, we have created an external GUI using python so that consumers can operate all the data files which we've created, from a single space rather than searching and opening them one by one exclusively. So it makes the implementation quite easy and user friendly as compared to other available interfaces which we could have implemented.

I. Literature Review

This paper [1] describes and have created different 3-D paper models using relatively complex form and they have successfully implemented complex computer mesh models and it has also been discussed about how to create folded mesh models and different text by constructing each letter individually. The limitation found in this paper was varieties could not be created implying the same method.

The idea of creating a rigged model was taken from paper[2] which has discussed about creating a open grid render farm and to develop a cost effective virtual reconstructions.. . this grid based render fam is an affordable GUI to manage different works related to farm.

Some of the features which have been added in my rigged 3-D model was asopted from paper [3] which discuses about eastimation of the facial movements . this project uses domain classifie to train the model.firstly they have trained the model using synthetic data and later with the real data which gave better results then the previous one.

The paper [4] has proposed project which is identified with AR and VR technique MICC utilizes surmising of inferred information in connecting 3D illustrations with a self-assertive application or area for which 3D substance is being made.

II. Methodology/Experimental

a. Tools and technologies

In tools and technologies, As you might know that Blender is a free as well as an open source 3D creation suite. So as after studying it thoroughly I noticed that it supports the entirety of the multidimensional modeling and animation, creation, rigging, , rendering, simulation, compositing as well as motion tracking, ll along with video editing and 2D animation interfaces. And so to say, about OpenCV that it is a cross-platform library applying which we can develop some real-time computerized projects which deals with real time inputs from the user, through webcams or microphone whatsoever it be. So OpenCV, it mainly focuses upon image processing, video capturing along with analysis which includes features like face detection as well as object detection. In this project I have used open cv to detect the facial movements. Python – For rendering different poly models the programming language is python.

b. Synthesis/Algorithm/Design/Method

Step 1) Firstly I Installed all the libraries, modules and dependencies which were required, then I followed with Blender, which has an embedded Python interpreter present already in it's bootloader, and the same interpreter, which is loaded only when Blender is started and that particular interpreter stays active all the time you are running the application. Now basic property of this interpreter is that it deals with scripts to draw

the user interface and is also useful for some of the internal tools comprised in blender as well. So along with the inbuilt dependencies, blender provides its Python modules for usage and application on such models, such as "bpy" and "mathutils", these are the modified versions of original modules which we use in python, to the interpreter which is pre embedded, so they can also be imported into a particular script and give an access to blender's data, classes, dependencies, functions, etc.
So this is a sample code snippet regarding how I got started keeping into consideration one particular model at the beginning:

import bpy
#(you can import it in the command prompt as well)

bpy.data.objects["Cube"].data.vertices[0].co.x += 1.0

#creating a basic foundation for the simple cube providing the coordinates for it's vertices.

Step 2) write subsequent code for the object we're supposed to create in our interface, we can create any number of objects as per our requirement and save them to our directory, the script of those objects which is written in python is saved alongside the model, so we can load the same model using the same script again in any other device, provided that we have pre-existing script which is runnable and complete in our default interface.

To run pre-existing script these are what the procedures we have to follows:

- Firstly load the particular script in the text editor or any other similar software and then select it then press "Run Script" option.
- You can also do one more thing that is just type it or paste it into the interactive console where we write our scripts all the time.
- And lastly to execute a python file (.py in extension)go to the command line then from the command line with Blender do the basic step for extraction as we usually do in windows console, for e.g: Blender _ myusername/program files/python/examplescript.py,

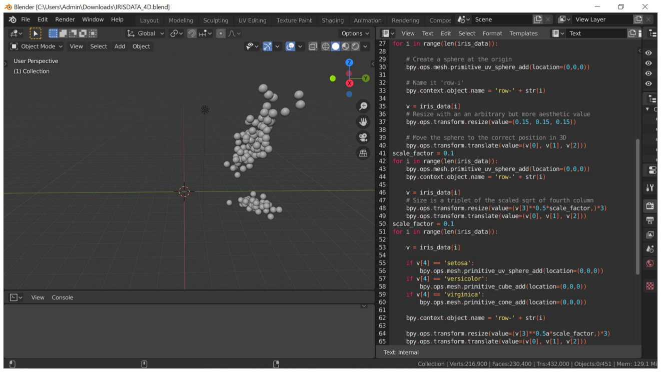
So this is basically the address of the program files which you store at a single place , you can obviously change it according to your own convenience.

Step 3) we save those files at a single place in our directory and then access them using our GUI, thus GUI we created is the frontline interface for our project and we can access all the files we created via here and to make things more convenient we can rename the models according to our requirement as well.

c. Table on the involved phases within the project.

Phase No.	**Overview**	**Implementations**	**Technicalities.**
1.	Mesh structures and Supershapes.	The desired graphical entities were generated.	***Using Bpy, mathutils, python, gpu-shader, etc.***
2.	External Model	The Vincent Model is appended which can now be controlled with the help of browser webcam IP.	***Used Bpy, mathutils, python, gpu-shader, etc. To create the model and then using OpenCV and HTML appended the same on site for user gesture control through real time inputs.***
3.	File Handling UI	A batch file explorer was implemented as a GUI for the user through which administrative control over all created mesh objects was awarded.	***Used tkinter, a Python library to make GUI for our app and can be used as a final output interface.***

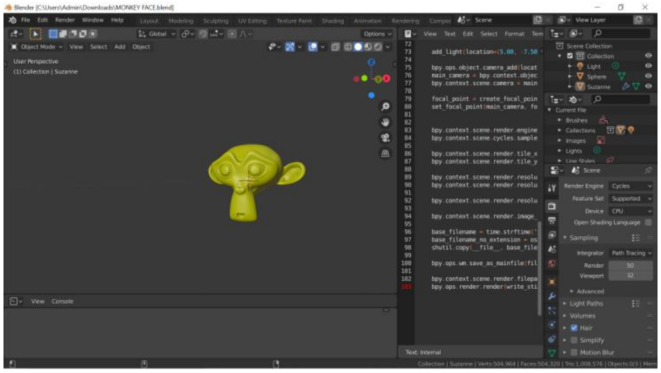
III. *DIAGRAMS/ FIGURES*



```
for i in range(len(iris_data)):

    # Create a sphere at the origin
    bpy.ops.mesh.primitive_uv_sphere_add(location=(0,0,0))

    # Name it 'row-i'
    bpy.context.object.name = 'row-' + str(i)

    v = iris_data[i]
    # Resize with an an arbitrary but more aesthetic value
    bpy.ops.transform.resize(value=(0.15, 0.15, 0.15))

    # Move the sphere to the correct position in 3D
    bpy.ops.transform.translate(value=(v[0], v[1], v[2]))
scale_factor = 0.1
for i in range(len(iris_data)):
    bpy.ops.mesh.primitive_uv_sphere_add(location=(0,0,0))
    bpy.context.object.name = 'row-' + str(i)

    v = iris_data[i]
    # Size is a triplet of the scaled sqrt of fourth column
    bpy.ops.transform.resize(value=(v[3]**0.5*scale_factor,)*3)
    bpy.ops.transform.translate(value=(v[0], v[1], v[2]))
scale_factor = 0.1
for i in range(len(iris_data)):

    v = iris_data[i]

    if v[4] == 'setosa':
        bpy.ops.mesh.primitive_uv_sphere_add(location=(0,0,0))
    if v[4] == 'versicolor':
        bpy.ops.mesh.primitive_cube_add(location=(0,0,0))
    if v[4] == 'virginica':
        bpy.ops.mesh.primitive_cone_add(location=(0,0,0))

    bpy.context.object.name = 'row-' + str(i)

    bpy.ops.transform.resize(value=(v[3]**0.5a*scale_factor,)*3)
    bpy.ops.transform.translate(value=(v[0], v[1], v[2]))
```

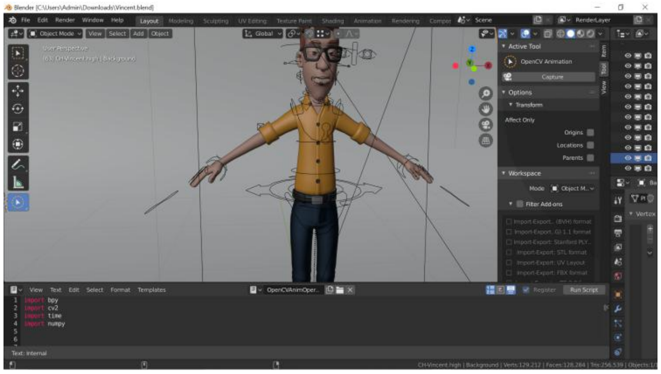
Blender [C:\Users\Admin\Downloads\IRISDATA_4D.blend]
User Perspective
(1) Collection
Text: Internal
Collection | Verts:216,900 | Faces:230,400 | Tris:432,000 | Objects:0/451 | Mem: 129.1 Mi

FIG (1). Graphing and Analyzing **Iris Data**



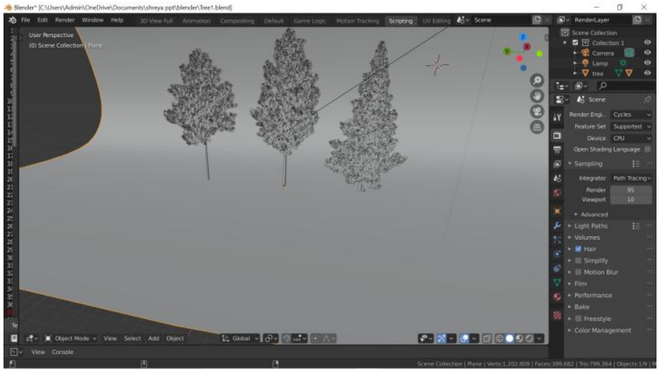
```

    add_light(location=(5.00, -7.50

    bpy.ops.object.camera_add(locat
    main_camera = bpy.context.objec
    bpy.context.scene.camera = main

    focal_point = create_focal_poin
    set_focal_point(main_camera, fo


    bpy.context.scene.render.engine
    bpy.context.scene.cycles.sample

    bpy.context.scene.render.tile_x
    bpy.context.scene.render.tile_y

    bpy.context.scene.render.resolu
    bpy.context.scene.render.resolu

    bpy.context.scene.render.resolu

    bpy.context.scene.render.image_

    base_filename = time.strftime('
    base_filename_no_extension = os
    shutil.copy(__file__, base_file

    bpy.ops.wm.save_as_mainfile(fil

    bpy.context.scene.render.filepa
    bpy.ops.render.render(write_sti
```

Fig (2) Rendering Monkey Face



```
import bpy
import cv2
import time
import numpy
```

Fig(3) Controlling 3D character



Render Engi.. Cycles
Feature Set Supported
Device CPU
Open Shading Language
Sampling
Integrator Path Tracing
Render 95
Viewport 10
Advanced
Light Paths
Volumes
☑ Hair
□ Simplify
□ Motion Blur
Film
Performance
Bake
□ Freestyle
Color Management

Fig(4) Demo on rendering tress which can be added 3D Games as an example.

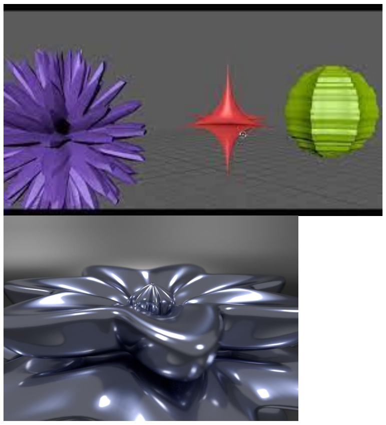


Fig.(5) Created supershapes being rendered in system.

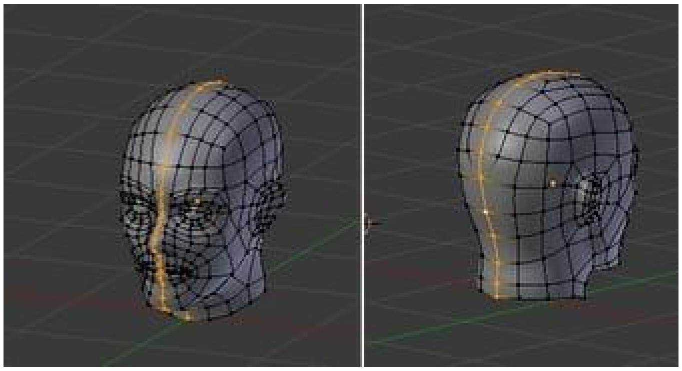


Fig.(6) Facial animation setup in Blender.

IV. Results and Discussions

We have managed to achieve our every objective in this project from creating 3D models to storing them as a seperate database for easier access. Thus we have our project which is significant for the usage regarding multidisciplinary projection of solid structures. Thus improvement in visualisation of multi dimensional data and spreading more clarity about the mesh structures and other minute details along with the basic geometry which the model may comprise is put into the light for study, in terms of the user perspective, as we can focus on every minute detail from external lightings to, applying gravity and see how it behaves while a freefall.

V. Limitations

The limitations of this projects is that AR cannot be integrated with games using blender as the software platform. Complex objects or models cannot be implemented with python scripts. Alongside the system requirements that must be met, viz. 64-bit dual core 2Ghz CPU with SSE2 support · 4 GB RAM · 1280×768 display · Mouse, trackpad or pen+tablet · Graphics card with 1 GB RAM, OpenGL 3.3 These are the minimum system requirements to run this project on your device along with latest version of Python and other necessary APIs.

VI. Future Scope

Further in this project of 3D Vincent model we can add some more animations to it and we can also render complex 3D shapes or build prototypes of different machines , laboratory equipment's or can develop a game using the same model eg. Endless runner games. There is a vast future scope for this prototype in the sector of education and industry, as there are no limits on the amount of data which we can handle, taken our memory bounds into consideration, also we can create any number of complex structures and try to alter as well as control them using real time inputs.

VII. Conclusion

For the concluding remarks we would like to prompt that all the previously set objectives of the project are met. Also we are successful in creating the GUI as well. Then one more major breakthrough was that we also appended the system for controlling our models too, rather than just creating them and showcasing in Cartesian space. So after adding the feature of gesture control with the help of OpenCV and different classifiers we have successfully implemented the real time controlling. Also speaking about the future scope, we have boundless benefits through this project as we have a vast area of application wherein we can generate almost every structure knowing how to code in python and then project the same, and also gesture controlling will be a huge break through here as we are able to do the real time input alters.

Acknowledgment

We would like to thank Professor Jyoti Madke and Professor Sangeeta Kurundkar from Vishwakarma Institute of Technology for guiding us throughout the completion of this project and motivating us to publish this idea.

References

I. https://www.irjet.net/archives/V7/i8/IRJET-V7I8582.pdf

II. https://ieeexplore.ieee.org/document/4839978

III. M. Z. Patoli; M. Gkion; A. Al-Barakati; W. Zhang; 2009 IEEE/PES Power Systems Conference and Exposition

IV. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9369299

V. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6860282

VI. The Blender Python (API Precision 3D Modeling and Add-On Development) Chris Conlan

VII. G. Bradski, A. Kaehler and O' Reilly, Learning OpenCV, 2008.

VIII. D. Stringhini, I.A. Souza, L.A. da Silva and M. Marengoni, Visão Computacional Usando OpenCV in Fundamentos de Visão Computacional, 2011.

IX. D. B. Kirk and W.W. Hwu, Programming Massively Parallel Processors: A Hands-on Approach, Morgan Kaufman, 2010.

X. NVIDIA CUDA C Programming Guide - 4.0, 2011.

XI. [online] Available: http://opencv.willowgarage.com/.

XII. J. Sanders and E. Kandrot, CUDA by Example - An Introduction to General-Purpose GPU Programming, Adison-Wesley, 2011.

XIII. OpenCV Reference Manual v2.2 December, 2010.

XIV. R.C. Gonzalez and R.E. Woods, Digital Image Processing, Prentice Hall, 2008.

XV. M. Sonka, V. Hlavac and R. Boyle, Image Processing Analysis and Machine Vision, Thomson, 2008.

XVI. J.P. Lewis, Fast Normalized Cross-Correlation, [online] Available: http://www.idiom.com/zilla/Papers/nvisionInterface/nip.html.

xvii. L. Breiman, J.H. Friedman, R.A. Olshen and C.J. Stone, Classification and Regression Trees, Belmont, CA:Wadsworth International, 1984.

xviii. L.I. Smith, A tutorial on Principal Components Analysis, [online] Available: http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf.

xix. M. Turk and A. Pentland, "Eigenfaces for Recognition", Journal of Cognitive Neurosicence, vol. 3, no. 1, pp. 71-86, 1991.

xx. T.M. Mitchell, Machine Learning, McGrawHill, 1997.

xxi. R. E. Kalman, "A new approach to linear filtering and prediction problems", Transactions of the ASME: Journal of Basic Engineering, vol. 82, pp. 35-45, 1960.

xxii. M. Isard and A. e Blake, "Condensation-conditional density propagation for visual tracking", International Journal in Computer Vision IJCV, vol. 29, no. 1, pp. 5-28, 1998.