Mathutils:

1. Cách Import?

from mathutils import Vector, Matrix

* Tất cả các tọa độ đỉnh, ma trận các thứ của Object trong Blender đều có kiểu Vector hoặc Matrix của Mathutils

1. Vector?

* Tạo 1 Vector

<Vector> = Vector(<Tọa Độ>)

* Ví dụ

foo = Vector((1, 2, 3))

* Tính tích có hướng của <Vector 1> với <Vector 2>

<Tích Có Hướng> = <Vectorr 1>.cross(<Vector 2>)

* Chuẩn hóa <Vector>

<Vector>.normalize()

* Ví dụ

foo.normalize()

* Ta có

|  |  |
| --- | --- |
| foo trước | [2, 4, 9] |
| foo sau | [0.199, 0.398, 0.896] |

* Trả về độ dài <Vector>

<Độ Dài> = <Vector>.length

1. Ma Trận?

* Tạo 1 ma trận

<Ma Trận> = Matrix(<Mảng 2D>)

* Ví dụ

foo = Matrix([

[1, 2, 3],

[5, 6, 7]

])

1. Toán Tử @?

* Nhân 2 ma trận hoặc 1 ma trận với 1 Vector

Context – Ngữ Cảnh:

1. Cách Import?

from bpy import context

1. Trả Về List Các Object Đang Được Chọn?

<List Các Object> = context.selected\_objects

1. Trả Về Object Mà Bạn Đã Chọn Trong Lần Chọn Gần Nhất?

<Object> = context.active\_object

Operation – Toán Tử:

1. Cách Import?

from bpy.ops import object as ob

1. Cách Chọn Và Bỏ Chọn Tất Cả Object?

ob.select\_all(action = <Kiểu Chọn>)

* Các cơ chế chọn tương ứng

|  |  |
| --- | --- |
| <Kiểu Chọn> | Cơ chế |
| "SELECT" | Chọn tất cả Object trong Scene hiện tại |
| "DESELECT" | Bỏ chọn tất cả Object trong Scene hiện tại |
| "TOGGLE" | "DESELECT" khi có ít nhất 1 Object được chọn, nếu không thì "SELECT" |
| "INVERT" | Chọn tất cả Object không được chọn và bỏ chọn tất cả Object được chọn trong Scene hiện tại |

Data – Dữ Liệu:

1. Cách Import?

from bpy import data

1. Trả Về Object Có Tên Nào Đó?

<Object> = data.objects[<Tên Object>]

* Trả về kiểu của <Object>

<Kiểu Object> = <Object>.type

* Các giá trị

|  |  |
| --- | --- |
| <Kiểu Object> |  |
| "MESH" | Các Object có đỉnh cạnh mặt các thứ |
| "CAMERA" | Cái Camera |
| "LIGHT" | Cái bóng đèn |

Object:

1. Mesh?

* Trả về Mesh của 1 Object

<Mesh> = <Object>.data

* Trả về 1 Data Collection chứa tất cả đỉnh của <Mesh>

<Các Đỉnh> = <Mesh>.vertices

* Trả về Vector tọa độ của 1 đỉnh theo hệ qui chiếu Local

<Tọa Độ Đỉnh Local> = <Đỉnh>.co

* Trả về trọng số Bevel của 1 đỉnh

<Trọng Số Bevel Đỉnh> = <Đỉnh>.bevel\_weight

* Trả về ma trận thế giới của Object, ma trận này có tác dụng chuyển vật từ hệ quy chiều Local sang Global

<Ma Trận Thế Giới> = <Object>.matrix\_world

* Dùng ma trận này để chuyển đổi hệ quy chiếu Local thành Global

<Tọa Độ Đỉnh Global> = <Ma Trận Thế Giới> @ <Tọa Độ Đỉnh Local>

Blender Mesh:

1. Cách Import?

import bmesh

1. Cách Hoạt Động?

* Copy 1 Mesh vào 1 Blender Mesh rồi chỉnh sửa rồi áp cái chỉnh sửa vào Mesh đã Copy
* Để tạo 1 Blender Mesh rỗng

<Blender Mesh> = bmesh.new()

* Để Copy 1 Mesh vào Blender Mesh

<Blender Mesh>.from\_mesh(<Mesh>)

* Để áp Blender Mesh vào lại Mesh

<Blender Mesh>.to\_mesh(<Mesh>)

* Để giải phóng Blender Mesh khỏi bộ nhớ

<Blender Mesh>.free()

Quick Code – Mã Nhanh:

1. Tích Tổng Moment Động Lượng?

from bpy import context

from mathutils import Vector, Matrix

from random import random

[obj] = context.selected\_objects

vertex\_masses = []

total\_mass = 0

center = Vector((0, 0, 0))

for vertex in obj.data.vertices:

vertex\_mass = random()

total\_mass += vertex\_mass

vertex\_masses.append(vertex\_mass)

center += vertex\_mass \* vertex.co

center = center / total\_mass + Vector(<Offset>)

omega\_direction = <Vector Vận Tốc Góc>

omega = Vector(omega\_direction)

normalized\_omega = Vector(omega\_direction)

normalized\_omega.normalize()

angular\_momentum = Vector((0, 0, 0))

distances = []

for idx, vertex in enumerate(obj.data.vertices):

r = vertex.co - center

d = r - normalized\_omega \* (r.dot(normalized\_omega))

m = vertex\_masses[idx]

angular\_momentum += m \* r.cross(omega.cross(d))

distances.append(d.length)

axis\_angular\_momentum = angular\_momentum.dot(normalized\_omega) \* normalized\_omega

print("Method 1 :")

print("Raw Angular Momentum : ", angular\_momentum)

print("Axis Angular Momentum : ", axis\_angular\_momentum)

moment\_of\_inertia = Matrix([

[0, 0, 0],

[0, 0, 0],

[0, 0, 0]

])

axis\_moment\_of\_inertia = 0

for idx, vertex in enumerate(obj.data.vertices):

x, y, z = vertex.co - center

m = vertex\_masses[idx]

moment\_of\_inertia += m \* Matrix([

[y\*\*2 + z\*\*2, -x\*y, -x\*z],

[-x\*y, x\*\*2 + z\*\*2, -y\*z],

[-x\*z, -y\*z, x\*\*2 + y\*\*2]

])

axis\_moment\_of\_inertia += m \* distances[idx]\*\*2

angular\_momentum = moment\_of\_inertia @ omega

axis\_angular\_momentum = axis\_moment\_of\_inertia \* omega

print("")

print("Method 2 : ")

print("Raw Angular Momentum : ", angular\_momentum)

print("Axis Angular Momentum : ", axis\_angular\_momentum)

print(moment\_of\_inertia)

* <Offset>, <Vector Vận Tốc Góc> là các Tuple gồm 3 phần tử là tọa độ
* Đoạn Code trên sẽ tính trọng điểm của Object đang chọn = cách coi nó là hệ chất điểm với các đỉnh là chất điểm với khối lượng ngẫu nhiên, lưu ý hệ quy chiếu Local, sau đó tìm mốc để tính Moment động lượng = trọng tâm + <Offset>
* Giả định rằng tất cả các đỉnh của Object đang quay quanh mốc với trục có hướng của <Vector Vận Tốc Góc>
* Đoạn Code trên cũng sẽ tính Moment động lượng = phương pháp nâng cao là sử dụng Moment quán tính, với mốc trùng với khi tính Moment động lượng và cũng in ra Moment quán tính
* Đoạn Code trên cũng sẽ trả về Moment động lượng theo trục quay = 2 phương pháp

1. Dùng Ma Trận Biến Đổi Hình Dạng Của 1 Object?

import bpy

from mathutils import Matrix

ob = bpy.context.object

me = ob.data

M = Matrix(<Ma Trận 4 x 4>)

me.transform(M)

me.update()

* <Ma Trận 4 x 4> là ma trận tích hợp quay và tịnh tiến bạn muốn dùng để biến đổi hình dạng Object
* Ví dụ

M = Matrix((

(1, 0, 0, 1),

(0, 1, 0, 0),

(0, 0, 1, 0),

(0, 0, 0, 1),

))

1. Plot Đồ Thị 3D Của Hàm Bất Kì?

* Đoạn code này xóa tất cả vật thể hiện tại, rồi tạo Object mới là đồ thị

import bpy

from math import \*

bpy.ops.object.select\_all(action='DESELECT')

bpy.ops.object.select\_by\_type(type='MESH')

bpy.ops.object.delete()

mesh = bpy.data.meshes.new("graph")

obj = bpy.data.objects.new("GraphObj", mesh)

scene = bpy.context.scene

scene.objects.link(obj)

vertices = []

edges = []

faces = []

num\_points = <Độ Phân Giải>

lim\_x = <Giới Hạn Giá Trị Của X>

lim\_y = <Giới Hạn Giá Trị Của Y>

x\_step = (lim\_x[1] - lim\_x[0]) / num\_points

y\_step = (lim\_y[1] - lim\_y[0]) / num\_points

for i in range(num\_points):

x = lim\_x[0] + i \* x\_step

for j in range(num\_points):

y = lim\_y[0] + j \* y\_step

z = <Biểu Thức Hàm>

vertices.append((x, y, z))

for i in range(num\_points - 1):

for j in range(num\_points - 1):

v0 = i \* num\_points + j

v1 = v0 + 1

v2 = v0 + num\_points

v3 = v2 + 1

edges.extend([(v0, v1), (v1, v3), (v3, v2), (v2, v0)])

faces.append((v0, v1, v3, v2))

mesh.from\_pydata(vertices, edges, faces)

mesh.update()

* <Giới Hạn Giá Trị Của X> và <Giới Hạn Giá Trị Của Y> là List bao gồm 2 giá trị cận có tác dụng giới hạn hiển thị hàm số
* Ví dụ

lim\_x = [-5, 5]

lim\_y = [-10, 10]

* <Độ Phân Giải> càng lớn thì đồ thị càng mượt
* <Biểu Thức Hàm> là hàm bạn muốn Plot
* Ví dụ hàm của bạn là z = 2x + 4y2

z = 2 \* x + 4 \* y \*\* 2

1. Plot Đường Cong?

* Đoạn Code này sẽ xóa tất cả Object hiện tại sau đó tạo đường cong có dạng

z = f(x, y), trong đó y = g(x)

import bpy

import numpy as np

from math import pow

def create\_curve():

curve\_data = bpy.data.curves.new(name='My Curve', type='CURVE')

curve\_data.dimensions = '3D'

spline = curve\_data.splines.new(type='POLY')

x\_values = np.linspace(<Cận Trái>, <Cận Phải>, <Độ Phân Giải>).tolist()

y\_values = [<Biểu Thức Y Theo X> for x in x\_values]

z\_values = [<Biểu Thức Đường Cong> for x, y in zip(x\_values, y\_values)]

spline.points.add(len(x\_values) - 1)

for i, (x, y, z) in enumerate(zip(x\_values, y\_values, z\_values)):

spline.points[i].co = (x, y, z, 1)

obj = bpy.data.objects.new('My Object', curve\_data)

scene = bpy.context.scene

scene.objects.link(obj)

obj.select = True

scene.objects.active = obj

bpy.ops.object.select\_all(action='DESELECT')

bpy.ops.object.select\_by\_type(type='CURVE')

bpy.ops.object.delete()

create\_curve()

* <Cận Trái> và <Cận Phải> là giới hạn hiển thị đường cong theo x
* <Độ Phân Giải> càng cao thì đường cong càng mượt
* <Biểu Thức Đường Cong> là f(x, y)
* Ví dụ z = f(x, y) = 2x + 4y2

2 \* x + 4 \* y \*\* 2

* <Biểu Thức Y Theo X> là g(x)
* Ví dụ y = g(x) = 5x

5 \* x

1. Render Từng Khoảng Frame Rời Rạc?

import bpy

FRAME\_RANGES = [<Các Khoảng Frame>]

OUTPUT\_PATH = <Đường Dẫn Tới Thư Mục Lưu Ảnh>

for frame\_range in FRAME\_RANGES:

start\_frame, end\_frame = frame\_range

for frame\_number in range(start\_frame, end\_frame + 1):

bpy.context.scene.frame\_set(frame\_number)

bpy.context.scene.render.filepath = OUTPUT\_PATH + str(frame\_number)

bpy.ops.render.render(write\_still=True)

* <Các Frame Range> là các Tuple chứa 2 phần tử lần lượt là Start Frame và End Frame ứng với mỗi khoảng Frame
* Ví dụ

[(0, 10), (20, 30), (50, 70)]

* Chạy đoạn Code trên sẽ Render Scene hiện tại dưới nền theo cài đặt ở giao diện người dùng

1. Xóa Toàn Bộ Key Frame Liên Quan Đến Chuyển Động Của Tất Cả Object Được Chọn?

import bpy

SELECTED\_OBJECTS = bpy.context.selected\_objects

for obj in SELECTED\_OBJECTS:

obj.animation\_data\_clear()

1. Xuất Tọa Độ Của Tất Cả Các Đỉnh Trong Object Được Chọn Sang File CSV?

import bpy

CSV\_FILE = <Đường Dẫn Tuyệt Đối Tới File CSV Có Phần Mở Rộng>

VERTICES = [

bpy.context.object.matrix\_world @ x.co

for x in bpy.context.object.data.vertices

]

f = open(CSV\_FILE, 'w')

f.writelines([";".join([str(x) for x in coord]) + "\n" for coord in VERTICES])

f.close()

* Trong File CSV, mỗi dòng sẽ là 1 bộ 3 hoành, tung, cao độ ngăn cách nhau bằng dấu chấm phẩy, ví dụ

1.0;1.0;-1.0

1.0;-1.0;1.0

1.0;-1.0;-1.0