Ball Balancer

TEAM BALL-BALANCER

September 2020

1 Introduction

1.0.1 Aim

To design a control system for the ledge to automatically balance the ball while a human tries to make the ball fall off using keyboard controls.

1.0.2 Basic idea of the project

The title of the project is a ball balancer. It's a game in which we have to design a control system for the ledge to balance the ball automatically. We will control the ball using the keyboard, and the PID controller controls the ledge. The game starts with the collision of the ball with the ledge. After that ball moves in the left and right direction on the ledge by pressing keys on the keyboard, and in response to the ball's movement, the ledge tries to rotate itself, so that ball remains on the ledge.
So, this is the basic idea of our project.

2 Virtual Environment

What is a Virtual Environment? At its core, Python virtual Environment's main purpose is to create an isolated environment for Python projects. This means each project can have its own dependencies, regardless of what dependencies every other project has.

2.1 Conda Environment

A conda environment is a directory that contains a specific collection of conda packages that you have installed. Conda effectively combines the functionality of pip and virtual env in a single package. When you begin using conda, you already have a default environment named base. You don't want to put programs into your base environment, though.

To create an environment with conda:

Open the terminal window (Anaconda prompt)
i. Check conda is installed and in your PATH.

ii. Check conda is up to date.

iii. Create a new environment for your project.

iv. Activate your environment.

v. Install additional python packages which you find necessary in your project.

vi. Deactivate your environment.

vii. Delete no longer needed environment.

There are some commands for create a new environment, to install packages and to manage environment:

Task	**Command**
Verify whether conda is installed, check version number	conda info
Update conda to the current version	conda update conda
Install a package included in Anaconda	conda install PACKAGENAME
Run a package after install, example Spyder*	spyder
Create a new environment named py35, install Python 3.5	conda create –name py35 python=3.5
Activate the new environment to use it	WINDOWS: activate py35 LINUX, macOS: source activate py35
Get a list of all my environments, active environment is shown with *	conda env list
List all packages and versions installed in active environment	conda list
Use conda to search for a package	conda search PACKAGENAME
Install a new package (Jupyter Notebook)in the active environment	conda install jupyter
Run an installed package (Jupyter Notebook)	jupyter-notebook
Install a package directly from PyPI into the current active environment using pip	pip install boltons
Remove one or more packages (toolz, boltons) from a specific environment (bio-env)	conda remove –name bio-env toolz boltons
Show version information for the current active Python	python –version

Table 1: environment management

3 Python Basic

Python is an object-oriented, structure-oriented, high-level, general-purpose programming language. Python is very useful in many aspects like Flexibility, Platform independence, Readability, an excellent library ecosystem.

Python is a straightforward, interactive programming language. Here is simple python code

o For input and print

```
# Python program showing
# a use of input()
val = input(”Enter your value: ”)
print(val)
Output:
Enter your value:123
123
```

o For iteration:

1)
```
for i in range(0, 10):
    print i
```
The output is : 0 1 2 3 4 5 6 7 8 9

2)
```
L = [1, 4, 5, 7, 8, 9]
```

```
for i in L:
print i
```

The output is : 1 4 5 7 8 9

3.0.1 numpy library

There is library NumPy for solving different problems based in an easy way. Here is one problem,
Question: You need to take an integer from the user, make an array of 10 random integers between zero and the number entered by the user, and print the numbers which are greater than 5.
PS: You have to do this assignment WITHOUT USING ANY LOOP anywhere in the code.
Here is logic:

```
import numpy as np
x=input(”enter the number”)
random=np.random.randint(0,x,10)
print(”random numbers between 0 and”,x)
print(random)
#printing elements above 5
print(”elements above 5”)print(random[random> 5])
```

4 Pygame

4.1 Workflow

1. pygame.init() [Initialization]

2. Game logic

3. pygame.quit() [Uninitiate]

The Game loop keeps our game open; otherwise, it will end in fractions of a second. Event loop checks for user input.
There are two kinds of surfaces
1)Display Surface: Only one and will always be shown
2)Regular surface or surface: It can be multiple and is only displayed when attached to the display surface.
Origin in pygame is the top left of our screen, and the cartesian system is as shown in the figure.

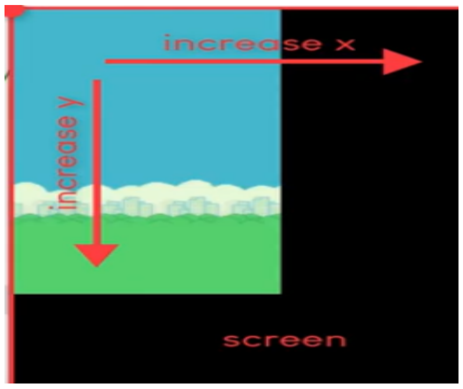

Figure 1: origin in pygame

Rect: It can be used to check collisions between our desired objects/images. It is an invisible frame in the output. It has various points that can be placed, as shown in the figure.

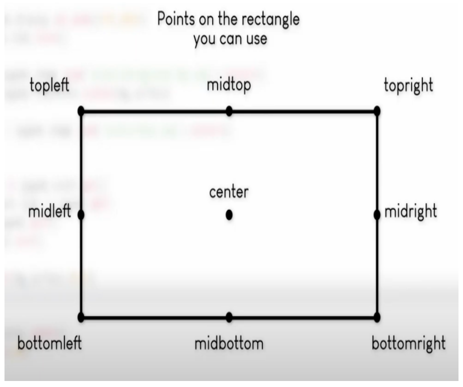

Figure 2: points on rectangle

All the below points do not place the rect but only move in one direction or axis and move the whole rectangle.

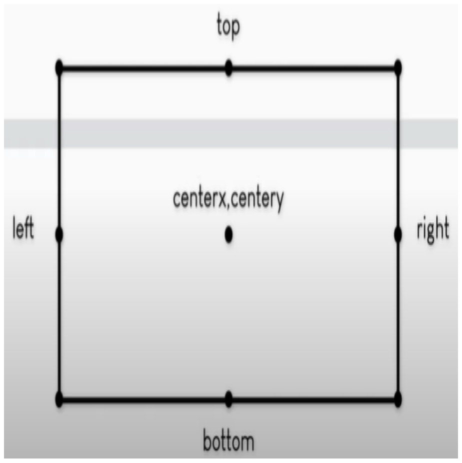

Figure 3: Points for movement of the rect

4.1.1 Functions Used

- Screen = pygame.display.set_mode((x,y))
  x and y are the pixel of the screen in which our game will run.Screen is a variable given by us.
- pygame.display.update()
  In the game loop this function and draws on the screen anything above it.
- event in pygame.event.get()
  This is used for all events in game i.e. input from the user.
  1. if event.type == pygame.QUIT:
     pygame.quit()
     It helps users to quit the game with the X button on top right.
  2. event.type == pygame.KEYDOWN
     if event.key == pygame.K_SPACE
     This registers the Space bar as an input key from the user.
  3. SPAWNPIPE = pygame.USEREVENT [SPAWNPIPE is a variable]
     pygame.time.set_timer(SPAWNPIPE,t)
     t is in milliseconds.
     This is not user defined but our own defined function with repeated rect occurring with time delay t ms.
- clock= pygame.time.Clock() helps to set frame rate or maximum frame rate of our game using clock.tick(max fps we want) eg=120. clock is a variable.
- backsurface = pygame.image.load('file location and name') is used to add pictures in the background where the backsurface is just a variable. It will not be seen on the screen if it is not present in our game loop. It can be implemented by using screen.blit().
  screen.blit(backsurface,(x,y))
  x,y are the coordinates as mentioned above.

backsurface=pygame.transform.scale2x(backsurface) magnifies our image 2 times.
.convert() at the end of the import line of images helps to increase the efficiency of pygame.

- Rect can be created by 2 ways

  1. pygame.Rect(width,height,x,y)
  2. surface.get_rect(rect_position)
     rect_position defines the place where it will be placed and then their coordinates. Eg
     center=(100,512)

  2nd one is preferred because it will get the surface and then place the rectangle around it so that this rectangle receives the same surface dimensions.
  pygame.transform.flip(variable,False/True,False/True)
  First, Boolean is for x-direction and the second one for the y-direction.
  x_rect.colliderect(y): checks for collision where x and y are variables.

- y=pygame.transform.rotozoom(surface,angle of rotation,x) rotozoom can scale(x times multiply) and rotate the surface where y is variable.

- For creating font
  y= pygame.font.Font('style',size) where y is variable
  For rendering text
  z= y.render('Value you want on screen',True/False,(Red value,Green value,Blue value))
  Each color value can be chosen from 0 to 255, which decides each color's intensity. True and false values represent sharpness of text.Value that you want on screen' should be a string, not an integer. If we're going to add text in the render function, we have to use f string used as
  (f' Text:(Value you want on screen))

- For adding sound we can use y = pygame.mixer.Sound('file location')
  For playing sound y.play()
  pygame.mixer.pre_init(frequency,size,channels,buffer) is added before pygame.pre_init() which tells us that we can initiate the mixer by changing frequency,size,channels and buffer.

- y = pygame.mask.from_surface(x) where x and y are variables.This function creates images using bits 1 represents part of image as white and 0 represents the null or black part as shown in figure.

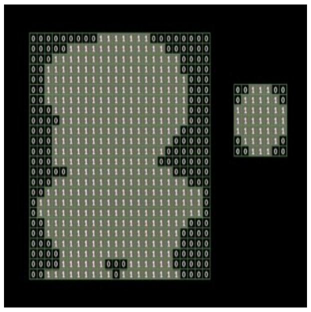

Figure 4: Masking of surface

- For the user to give input from the mouse we use.
  y = pygame.mouse.get_pos()

- y= x_overlap(z,offset value). This function helps to identify where the pixel of x overlaps with z's pixel offset value determines whether the rectangle has collided with the other specified rectangle.

4.2 Ledge Rotation

After setting the background, we add a ledge on the screen. We need to rotate the ledge for the game.

- def rotate(surface,angle):
  rotated_surface = pygame.transform.rotozoom (surface,angle,1)
  rotated_rect = rotated_surface.get_rect(center = (450,350))
  return rotated_surface,rotated_rect

- Here, rotozoom is used for rotation of ledge. In function surface and angle is mentioned for rotation. When the ledge starts rotating, at that time angle = 0, and then angle increases.

- ledge_rect = ledge.get_rect(center = (450,350))
  angle = 0
  angle += 1
  rotated_ledge, rotated_ledge_rect = rotate(ledge,angle)
  screen.blit(rotated_ledge,rotated_ledge_rect)
  This way, by making function we can code for ledge rotation.

- ledge_rect = ledge.get_rect(center = (450,350))
  angle = 0
  angle += 1
  rotated_ledge = pygame.transform.rotate(ledge,angle)
  rotated_ledge_rect = rotated_ledge.get_rect(center = (450,350))
  screen.blit(rotated_ledge,rotated_ledge_rect)
  Here, ledge is mentioned for rotation and angle is mentioned for at which angle, ledge is rotated.

5 Pymunk

Pymunk can be used to calculate physics, and pygame is used for its visualizing physics. In pymunk, the space is where the physics is being calculated, which is continuously updated; otherwise, it is just standing still. Inside of space, we can create physical objects which can be made by the body. A body is a thing that can accept physics but does not have a shape, but it can be specified.

- space = pymunk.Space() //helps us to create the space (space is a variable)
  space.gravity=(v,u) where v and u are horizontal and vertical gravity.
  space.step(number on how fast we need to update, i.e., 1/FPS(frames per second)) is used to update our simulation.

- Every time we update our screen, we also update our simulation.
  pymunk.Body() creates a body

- Body can be of types
  1)Static body(does not move but can collide with others)
  2)Dynamic body (React with objects and moves by physics)
  3)Kinematic body(can be moved by player or physics)

- For a dynamic body there are 3 parameters
  Mass(Weight),Inertia,body_type
  pymunk.Body(mass,inertia,body_type = pymunk.Body.DYNAMIC)
  body.position = (x,y) where body is the variable and x,y are the coordinates

- Body is like atom that doesn't have shape so it can't perform collision so to add shape to this body we use ...
  shape = pymunk.Circle(body,radius) creates a circle
  Shape=pymunk.Poly.create_box(body,(l,w)). //creates a box(rectangular) having length l and width w.
  space.add(body,shape) adds this to our physics simulation. (shape and body are variables).
  Joint=pymunk.constraint.PivotJoint(space.static_body,body,(x,y)) where x and y are the coordinates of hinge point. A pivot joint allows two objects to pivot about a single point.

- Now to rotate the body with the constant angular velocity we use
  Joint_motor=pymunk.constraint.SimpleMotor(space.static_body,body,-5) //where -5 is the rate i.e desired angular velocity.

- We restricted the maximum force that constraint can use to act on two bodies to 500000000 by using - Joint_motor.max_force =500000000.
  We took both ball and ledge as a dynamic object in our object; our window is added as a static object.

6 PID

- Input in the plant is the actuating signal which can give us the desired output or controlled variable. Feedback control compares the command to see how far off the system is from where we want it.This difference between the two is the error term, and we want to reduce it to zero, which can be done by PID controller.

- PID controller is called proportional integral derivative.In the proportional control algorithm, the controller output is proportional to the error signal, which is the difference between the setpoint and the process variable. In other words, the output of a proportional controller is the multiplication product of the error signal and the proportional gain.

- Derivative produces a measure of the rate of change of the error, which is how fast the error is growing or shrinking. The I uses the past error, P uses the present error, and D uses the future.These three branches contribute to the controller's overall output, and for the designer, we can vary contributions to minimize error.These changes are called tuning of the controller.

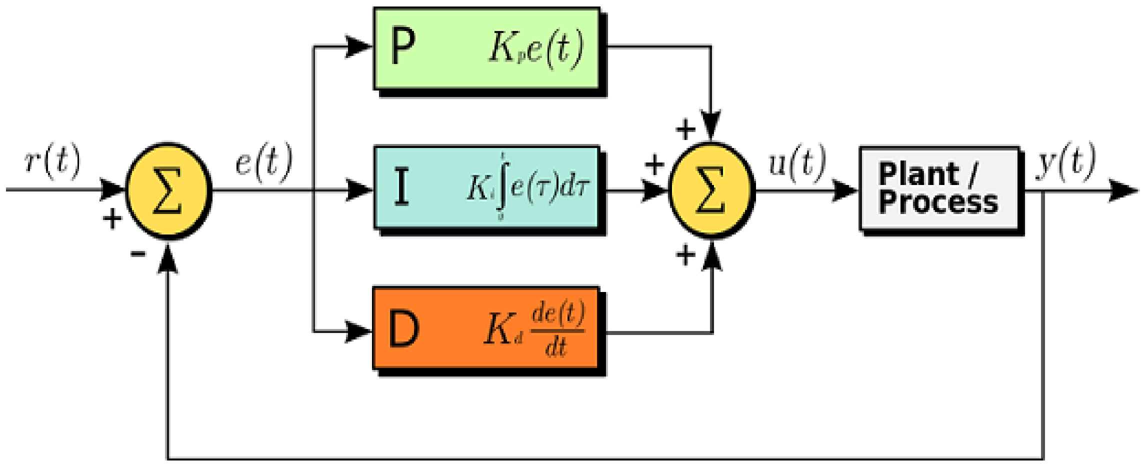

Figure 5: PID-workflow

7 Flowchart Of Logic

Here is a flowchart of logic, which we have implemented for ball balancing. Now, when the pygame window

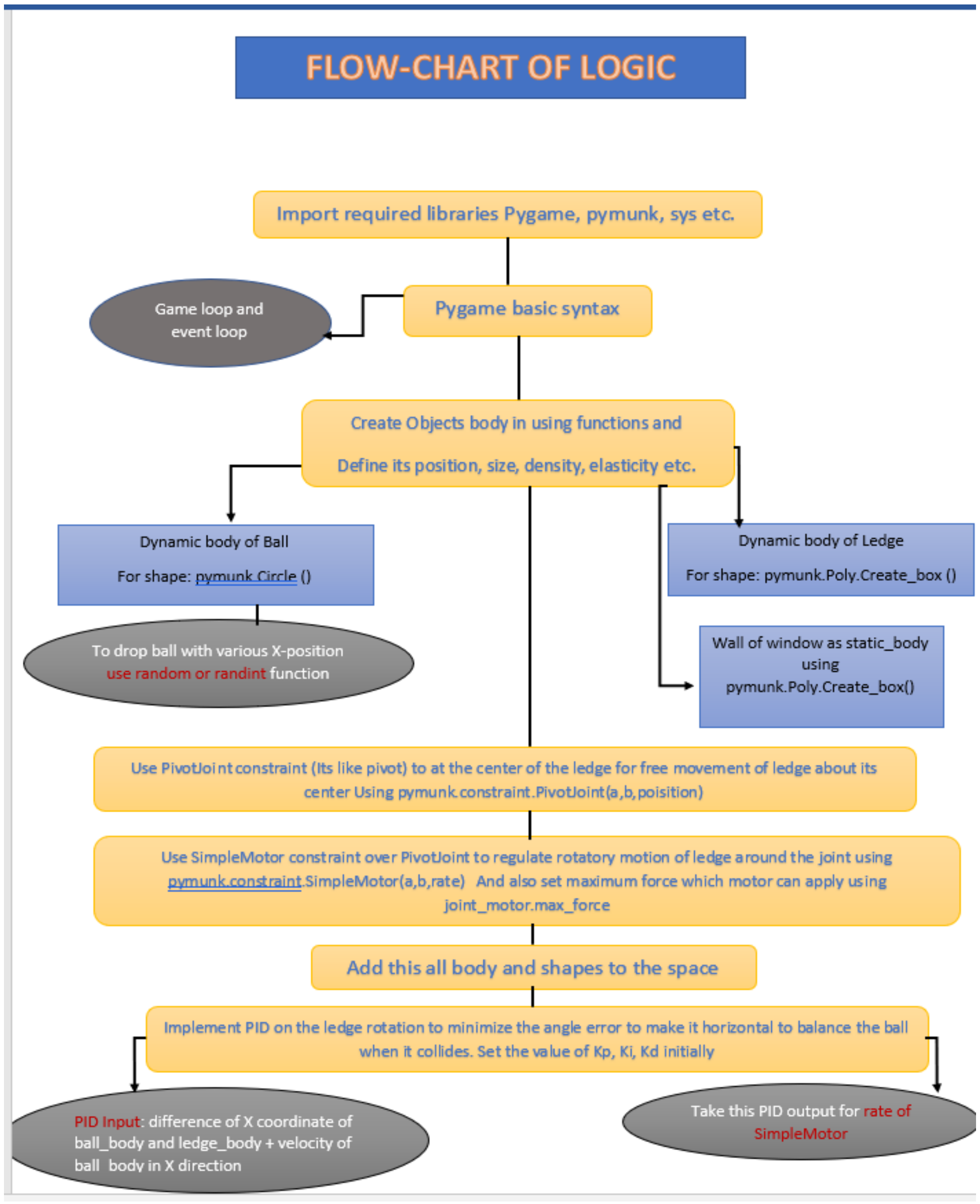
FLOW-CHART OF LOGIC
Import required libraries Pygame, pymunk, sys etc.
Pygame basic syntax
Game loop and event loop
Create Objects body in using functions and Define its position, size, density, elasticity etc.
Dynamic body of Ball For shape: pymunk.Circle ()
To drop ball with various X-position use random or randint function
Dynamic body of Ledge For shape: pymunk.Poly.Create_box ()
Wall of window as static_body using pymunk.Poly.Create_box()
Use PivotJoint constraint (Its like pivot) to at the center of the ledge for free movement of ledge about its center Using pymunk.constraint.PivotJoint(a,b,poisition)
Use SimpleMotor constraint over PivotJoint to regulate rotatory motion of ledge around the joint using pymunk.constraint.SimpleMotor(a,b,rate) And also set maximum force which motor can apply using joint_motor.max_force
Add this all body and shapes to the space
Implement PID on the ledge rotation to minimize the angle error to make it horizontal to balance the ball when it collides. Set the value of Kp, Ki, Kd initially
PID Input: difference of X coordinate of ball_body and ledge_body + velocity of ball body in X direction
Take this PID output for rate of SimpleMotor

Figure 6: Flowchart

starts, the ball will drop on the ledge, and the ledge will try to balance it. PID will minimize the angle's error to make the ledge horizontal.