PONG

- With a dash of AI

Overview

Problem Statement

Project Goals

The Implementation

Demo

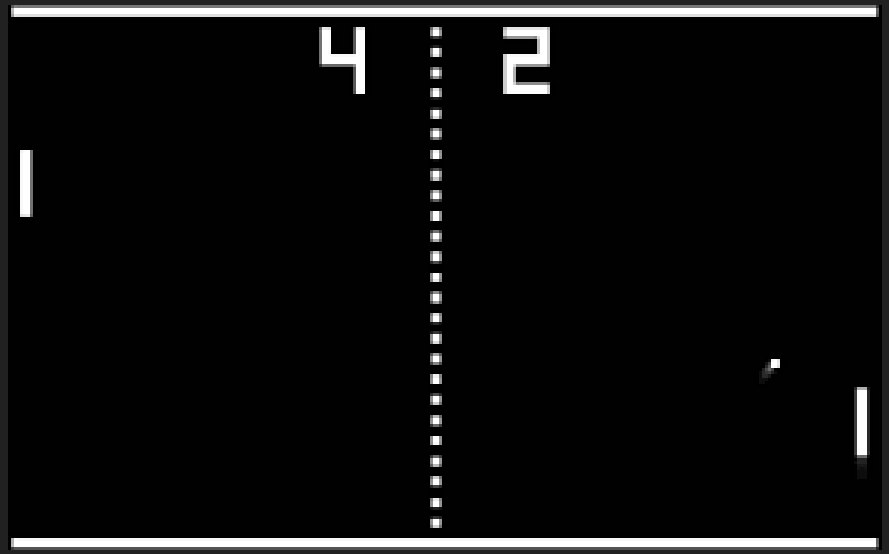
The Problem Statement

- To build a python-based app from scratch that plays pong
- Interfaces:

  Player vs Player

  Player vs Computer

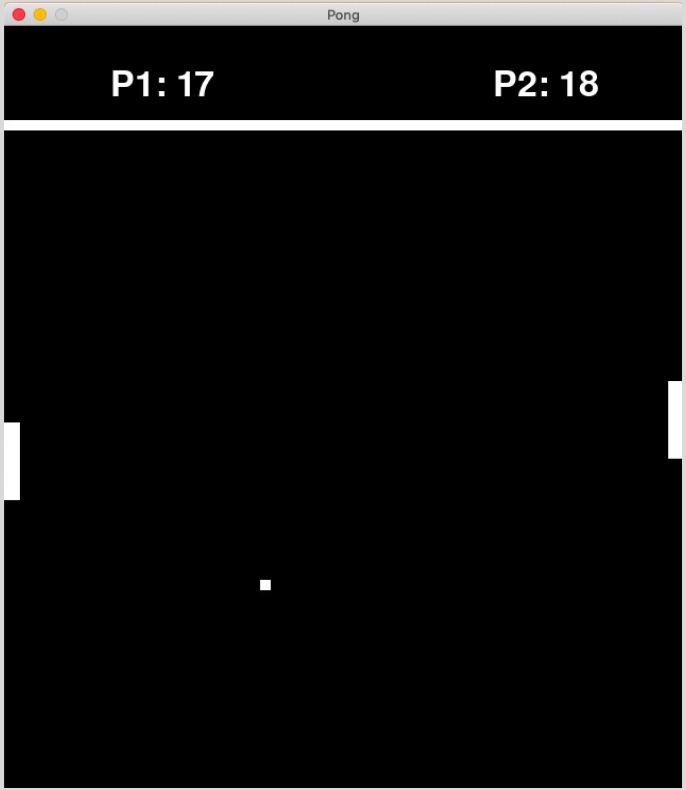
Something Just Like This



Project Goals

1. **Build a GUI**
2. **Create the game interface**
3. **Create AI models**
4. **Train the AI models**

The Implementation

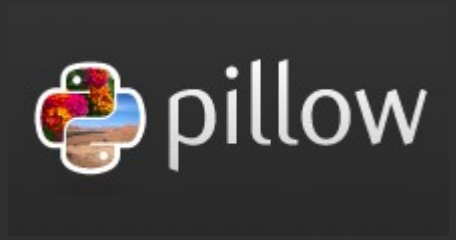


os

shutil

random

LIBRARIES



sys

math

importlib

Building the GUI

Reactions, text, buttons added using Thorpy

```python
# create a reaction that calls the function when space is pressed
space = thorpy.Reaction(reacts_to=pygame.KEYDOWN,
                        reac_func=spacedown,
                        event_args={"key": pygame.K_SPACE})

# escape quits the game
esc = thorpy.Reaction(reacts_to=pygame.KEYDOWN,
                      reac_func=escape,
                      event_args={"key": pygame.K_ESCAPE})      You, a few seconds

# initialize the window
application = thorpy.Application((640, 740), "Pong")

# create all elements
title = thorpy.make_text(text="PONG", font_size=50, font_color=(255, 0, 0))
instr_text = "Press Space to Play"
instr = thorpy.make_text(
    text=instr_text, font_size=30, font_color=(50, 100, 50))
elements = [title, instr]

# make a box to add the elements
box = thorpy.Box.make(elements=elements)
box.fit_children(margins=(30, 30))
box.center()
box.set_main_color((220, 220, 220, 180))

# generate background and add element box to it
background = thorpy.Background.make(
    color=(0, 0, 0), elements=[box])

# add reaction to window
background.add_reaction(space)
menu = thorpy.Menu(background)

# activate the objects
menu.play()
```

Creating the game interface

Key events, moving objects using Pygame

```
# control loop
while 1:

    # set frame rate
    clock.tick(60)
    # black screen
    screen.fill((0, 0, 0))
    #  set repeat rate for keys, press event registers in intervals of 100 ms only
    pygame.key.set_repeat(100, 100)

    # event loop
    for event in pygame.event.get():

        # exit on quit event
        if event.type == pygame.QUIT:
            # do something?
            return

        # key is pressed
        elif event.type == pygame.KEYDOWN:

            # move the left pad if w or s is pressed
            if event.key == pygame.K_w:
                player1.moveup()
            if event.key == pygame.K_s:
                player1.movedown()

            # move the right pad if down or up arrow key is pressed
            if event.key == pygame.K_UP:
                player2.moveup()
            if event.key == pygame.K_DOWN:
                player2.movedown()

        # pressed key is released
        elif event.type == pygame.KEYUP:

            # stop moving left pad when w or s is released
            if event.key == pygame.K_w or event.key == pygame.K_s:
                player1.movepos = [0, 0]
                player1.state = "still"
```

Creating the AI models

5 Models:

- Random movement
- 2 Ball following
- 2 RL based

```python
import os.path as path
from app import data


def make_prediction(ball_x, ball_y, pad_x, pad_y):

    if(ball_y < pad_y + 40):
        # move up if ball is 40 pixels below pad's centre
        pred = 1
        return pred

    elif(ball_y > pad_y + 40):
        # move down if ball is 40 pixels above pad's centre
        pred = -1
        return pred

    else:
        # don't move at all
        pred = 0
        return pred


def get_prediction(side):

    # get ball and pad positions
    ball_x, ball_y, angle, v = data.get_ball_pos()
    if(side == "left"):
        pad_x, pad_y = data.get_pad_pos("left")
    elif(side == "right"):
        pad_x, pad_y = data.get_pad_pos("right")

    # make and return action
    prediction = make_prediction(ball_x, ball_y, pad_x, pad_y)
    return prediction
```

Reinforcement Learning

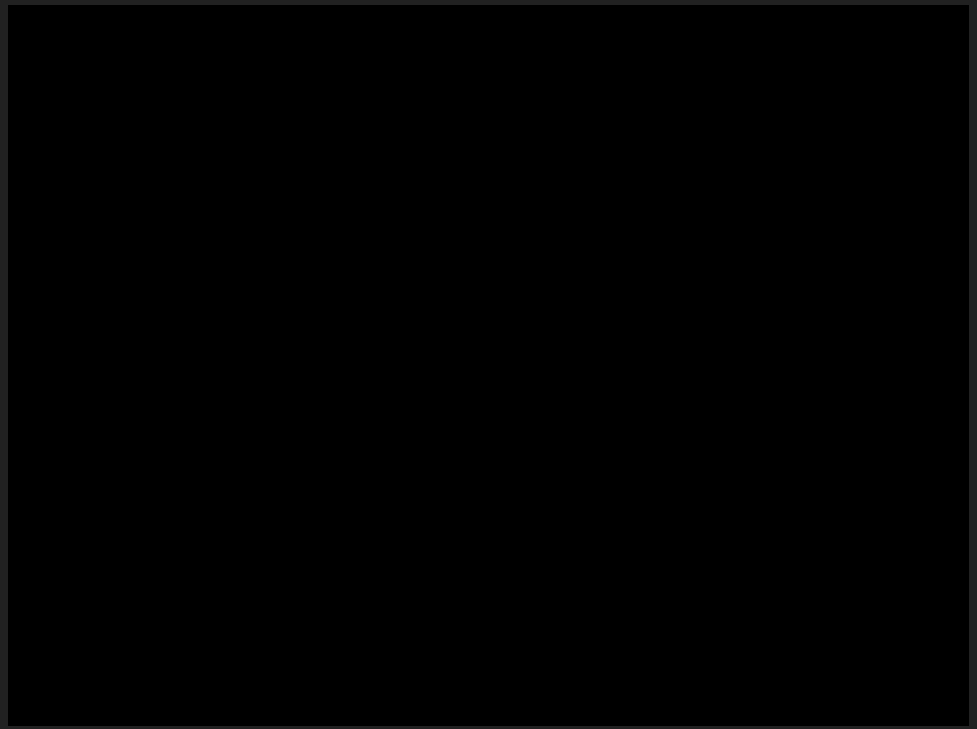
- Teaching the AI how to play by rewarding it for the correct action
- Can be done using different algorithms: Policy Gradients, Q-Learning

Policy Gradients:

1. Start with random actions
2. Play a few games
3. Collect rewards
4. Modify actions to maximize reward
5. Repeat

Training the AI models

Train the RL based AI models using the other AI models or with the player



DEMO

Next Steps

1. More training
2. More models
3. Convert to executables for smooth distribution across all systems

Questions?