AI PRESENTATION

Flappy bird

TEAM 21

- Phí Hoàng Long 20184288
- Vũ Tùng Lâm 20184283
- Nguyễn Văn Lực 20176812

FLAPPY BIRD

Introduction

- FLAPPY BIRD is a mobile game developed by a Vietnamese video game artist and programmer Dong Nguyen, under his game development company dotGears.

FLAPPY BIRD

Introduction

- FLAPPY BIRD is a mobile game developed by a Vietnamese video game artist and programmer Dong Nguyen, under his game development company dotGears.
- Released in May 2013 and at the end of January 2014, the most downloaded free game in Appstore.

FLAPPY BIRD

Introduction

- FLAPPY BIRD is a mobile game developed by a Vietnamese video game artist and programmer Dong Nguyen, under his game development company dotGears.
- Released in May 2013 and at the end of January 2014, the most downloaded free game in Appstore.
- Removed from Appstore and Google Play by its creator due to what he considered to be its addictive nature and overuse.

FLAPPY BIRD

Introduction

- A side-scrolling mobile game featuring 2D retro style graphics.

FLAPPY BIRD

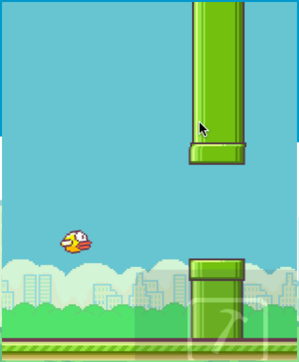
Introduction

- A side-scrolling mobile game featuring 2D retro style graphics.
- The objective is to direct a flying bird, name “Faby”.

FLAPPY BIRD

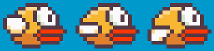
Introduction

- A side-scrolling mobile game featuring 2D retro style graphics.
- The objective is to direct a flying bird, name "Faby".



FLAPPY BIRD

Load images and set window

- Bird's status : 

FLAPPY BIRD

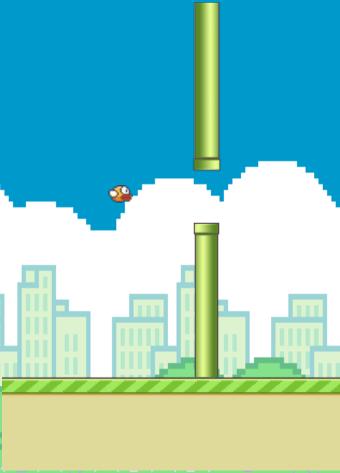
Load images and set window

- Bird's status :
- Install pygame, neat-python and remove neat library

  ```
  pip install pygame neat-python
  pip uninstall neat
  ```

```
        #import libraries
    import pygame
    import neat

```



FLAPPY BIRD

Load images and set window

- Load images:

```python
    #Load images
BIRD_IMGS = [pygame.image.load(os.path.join("img", "bird1.png")
    ),
    pygame.image.load(os.path.join("img", "bird2.png")),
    pygame.image.load(os.path.join("img", "bird3.png"))]
PIPE_IMG = pygame.transform.scale(pygame.image.load(
    os.path.join("img", "pipe.png")), (60, 400))
BASE_IMG = pygame.transform.scale(pygame.image.load(
    os.path.join("img", "base.png")), (400, 112))
BG_IMG = pygame.transform.scale(pygame.image.load(
    os.path.join("img", "bg.png")), (400, 600))
```

FLAPPY BIRD

Load images and set window

- Load images:

```
    #Load images
BIRD_IMGS = [pygame.image.load(os.path.join("img", "bird1.png")
    ),
    pygame.image.load(os.path.join("img", "bird2.png")),
    pygame.image.load(os.path.join("img", "bird3.png"))]
PIPE_IMG = pygame.transform.scale(pygame.image.load(
    os.path.join("img", "pipe.png")), (60, 400))
BASE_IMG = pygame.transform.scale(pygame.image.load(
    os.path.join("img", "base.png")), (400, 112))
BG_IMG = pygame.transform.scale(pygame.image.load(
    os.path.join("img", "bg.png")), (400, 600))

```

- Load windows:

```
    #Load window
WIN_WIDTH = 400
WIN_HEIGHT = 600
WIN = pygame.display.set_mode((WIN_WIDTH, WIN_HEIGHT))

```

FLAPPY BIRD

Evaluate a genome and draw window

- Evaluation function

```python
    # Evaluation function
def eval_genomes(genomes, config):
    for pipe in pipes:
        for index, bird in enumerate(birds):
            if pipe.collide(bird):
                ge[index].fitness -= 1
                birds.pop(index)
                nets.pop(index)
                ge.pop(index)
            if not pipe.passed and pipe.x < bird.x:
                pipe.passed = True
                add_pipe = True
        if pipe.x + pipe.PIPE_TOP.get_width() < 0:
            rem.append(pipe)
        pipe.move()
    if add_pipe:
        score += 1
        pipes.append(Pipe(400))
```

FLAPPY BIRD

Evaluate a genome and draw window

- Draw window

```python
    # Draw window
def draw_window(win, birds, pipes, base, score, GEN, pipe_ind):
    if GEN == 0:
        GEN = 1
    win.blit(BG_IMG, (0,0))
    for pipe in pipes:
        pipe.draw(win)
    base.draw(win)
    for bird in birds:
        bird.draw(win)
    pygame.display.update()
```

FLAPPY BIRD

Genetic Algorithm

Definition

Genetic algorithm Genetic algorithm is a search heuristic that is inspired by Charles Darwin's theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation

Genetic Algorithm implementation

1. Initialize the population :

```
        p = neat.Population(config)

```

FLAPPY BIRD

Genetic Algorithm

Definition

Genetic algorithm Genetic algorithm is a search heuristic that is inspired by Charles Darwin's theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation

Genetic Algorithm implementation

1. Initialize the population :

```
        p = neat.Population(config)

```

2. Create neural network for each unit :

```
    for _,g in genomes :
        g.fitness = 0
        net = neat.nn.FeedForwardNetwork.create(g, config)
        nets.append(net)
        birds.append(Bird(150,250))

```

FLAPPY BIRD

Genetic Algorithm

Genetic Algorithm implementation

3. Calculate the fitness-function :

```
        for x, bird in enumerate(birds):
            bird.move()
            ge[x].fitness += 0.1
            output = nets[x].activate((bird.y,
                abs(bird.y - pipes[pipe_ind].height),
                abs(bird.y - pipes[pipe_ind].bottom)))
            if output[0] > 0.5:
                bird.jump()

```

Genetic Algorithm implementation

3. Calculate the fitness-function :

```python
        for x, bird in enumerate(birds):
            bird.move()
            ge[x].fitness += 0.1
            output = nets[x].activate((bird.y,
                abs(bird.y - pipes[pipe_ind].height),
                abs(bird.y - pipes[pipe_ind].bottom)))
            if output[0] > 0.5:
                bird.jump()

```

4. Evaluate the current population for the next:

```python
        # population.py
        k = 0
        while n is None or k < n:
            k += 1
            self.reporters.start_generation(self.generation)
            ...

```

Thank you!