

目 录

名词

chatgpt使用

 chatgpt类似

 chatgpt-使用-0-贪吃蛇游戏

ollama

 [out-of-date]Windows10本地使用ollama和chatbox运行deepseek

AI绘图

 comfyUI

 ComfyUI安装与使用

 stable-diffusion-webui-getting-started

 stable-diffusion-webui

 sd-webui-and-SadTalker

 sd-webui-and-easyphoto

 sd-webui-cleaner

 sd-webui-text2video

 sd-webui-animatediff

 sd-webui-segment-anything

tts

 tts-stt-by-buzz

 edge-tts

 tts链接

ai常用链接

AI编程

 ai code generater

 opencode

 claude

多媒体工具

 FFmpeg 音频视频处理

名词

Language Model -> Large Language Model

对话只能一问一答，你的问话叫 Prompt

问话中的背景信息叫Context

每次对话时把之前的对话保留下，称为Memory。也就有了多次问答

用户问A一段话，A去上网搜索内容并返回，A是一段神秘代码，被称为智能体Agent。

Agent能上网搜索，当然也应该能搜索本地文档

Search: 向量数据库相关内容片段

检索增强生成 Retrieval-Augmented-Generation

Agent与LLM沟通时的约定：Function Calling

Agent调用搜索/文件读取/脚本执行/pdf转换等工具集时 使用的协议叫ModelContextProtocol

LangChain

workflow

固化一些提示词：

SKILL (是一个Prompt加载器，)

chatgpt使用

chatgpt类似

[ChatGPT 免费在线体验镜像网站](#)

[LiLittleCat/awesome-free-chatgpt](#): 免费的 ChatGPT 镜像网站列表，持续更新。List of free ChatGPT mirror sites, continuously updated.

chatgpt-使用-0-贪吃蛇游戏

贪吃蛇游戏

v1.0

我：请用pygame编写一个贪吃蛇的游戏，写出可以完整运行的代码

```
import pygame
import random

# 初始化pygame
pygame.init()

# 设置屏幕大小
WIDTH, HEIGHT = 600, 400
CELL_SIZE = 20
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("贪吃蛇")

# 颜色定义
WHITE = (255, 255, 255)
GREEN = (0, 255, 0)
RED = (255, 0, 0)
BLACK = (0, 0, 0)

# 方向定义
UP = (0, -1)
DOWN = (0, 1)
LEFT = (-1, 0)
RIGHT = (1, 0)

# 初始化蛇和食物
snake = [(100, 100), (90, 100), (80, 100)]
direction = RIGHT
food = (random.randint(0, (WIDTH // CELL_SIZE) - 1) * CELL_SIZE,
        random.randint(0, (HEIGHT // CELL_SIZE) - 1) * CELL_SIZE)

# 运行标志
running = True
clock = pygame.time.Clock()
```

```
while running:
    screen.fill(BLACK)

    # 事件监听
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP and direction != DOWN:
                direction = UP
            elif event.key == pygame.K_DOWN and direction != UP:
                direction = DOWN
            elif event.key == pygame.K_LEFT and direction != RIGHT:
                direction = LEFT
            elif event.key == pygame.K_RIGHT and direction != LEFT:
                direction = RIGHT

    # 计算新蛇头位置
    new_head = (snake[0][0] + direction[0] * CELL_SIZE, snake[0][1] + direction[1] * CELL_SIZE)

    # 判断是否撞墙或撞到自己
    if new_head in snake or new_head[0] < 0 or new_head[1] < 0 or new_head[0] >= WIDTH or new_head[1] >= HEIGHT:
        running = False
        break

    # 添加新头
    snake.insert(0, new_head)

    # 判断是否吃到食物
    if new_head == food:
        food = (random.randint(0, (WIDTH // CELL_SIZE) - 1) * CELL_SIZE,
                random.randint(0, (HEIGHT // CELL_SIZE) - 1) * CELL_SIZE)
    else:
        snake.pop()

    # 画蛇
    for segment in snake:
        pygame.draw.rect(screen, GREEN, (segment[0], segment[1], CELL_SIZE, CELL_SIZE))

    # 画食物
    pygame.draw.rect(screen, RED, (food[0], food[1], CELL_SIZE, CELL_SIZE))

    # 刷新屏幕
```

```
pygame.display.flip()
clock.tick(10)

pygame.quit()
```

总结：生成的代码可直接运行，但是碰到墙就结束。

v1.1

我：优化一下，使用类和main调用来运行游戏

```
import pygame
import random

class SnakeGame:
    def __init__(self):
        pygame.init()
        self.WIDTH, self.HEIGHT = 600, 400
        self.CELL_SIZE = 20
        self.screen = pygame.display.set_mode((self.WIDTH, self.HEIGHT))
        pygame.display.set_caption("贪吃蛇")

        self.WHITE = (255, 255, 255)
        self.GREEN = (0, 255, 0)
        self.RED = (255, 0, 0)
        self.BLACK = (0, 0, 0)

        self.UP = (0, -1)
        self.DOWN = (0, 1)
        self.LEFT = (-1, 0)
        self.RIGHT = (1, 0)

        self.snake = [(100, 100), (90, 100), (80, 100)]
        self.direction = self.RIGHT
        self.food = self.generate_food()

        self.running = True
        self.clock = pygame.time.Clock()

    def generate_food(self):
        return (random.randint(0, (self.WIDTH // self.CELL_SIZE) - 1) * self.CELL
        _SIZE,
                random.randint(0, (self.HEIGHT // self.CELL_SIZE) - 1) * self.CEL
        L_SIZE)
```

```
def handle_events(self):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            self.running = False
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP and self.direction != self.DOWN:
                self.direction = self.UP
            elif event.key == pygame.K_DOWN and self.direction != self.UP:
                self.direction = self.DOWN
            elif event.key == pygame.K_LEFT and self.direction != self.RIGHT:
                self.direction = self.LEFT
            elif event.key == pygame.K_RIGHT and self.direction != self.LEFT:
                self.direction = self.RIGHT

def update(self):
    new_head = (self.snake[0][0] + self.direction[0] * self.CELL_SIZE,
                self.snake[0][1] + self.direction[1] * self.CELL_SIZE)

    if new_head in self.snake or new_head[0] < 0 or new_head[1] < 0 or \
        new_head[0] >= self.WIDTH or new_head[1] >= self.HEIGHT:
        self.running = False
        return

    self.snake.insert(0, new_head)

    if new_head == self.food:
        self.food = self.generate_food()
    else:
        self.snake.pop()

def draw(self):
    self.screen.fill(self.BLACK)
    for segment in self.snake:
        pygame.draw.rect(self.screen, self.GREEN, (segment[0], segment[1], self.CELL_SIZE, self.CELL_SIZE))
    pygame.draw.rect(self.screen, self.RED, (self.food[0], self.food[1], self.CELL_SIZE, self.CELL_SIZE))
    pygame.display.flip()

def run(self):
    while self.running:
        self.handle_events()
        self.update()
        self.draw()
        self.clock.tick(10)
    pygame.quit()
```

```

if __name__ == "__main__":
    game = SnakeGame()
    game.run()

```

v1.2

我：设置慢、中、快三种速度，默认是慢，用户可以在菜单里修改

```

import pygame
import random

class SnakeGame:
    def __init__(self):
        pygame.init()
        self.WIDTH, self.HEIGHT = 600, 400
        self.CELL_SIZE = 20
        self.screen = pygame.display.set_mode((self.WIDTH, self.HEIGHT))
        pygame.display.set_caption("贪吃蛇")

        self.WHITE = (255, 255, 255)
        self.GREEN = (0, 255, 0)
        self.RED = (255, 0, 0)
        self.BLACK = (0, 0, 0)

        self.UP = (0, -1)
        self.DOWN = (0, 1)
        self.LEFT = (-1, 0)
        self.RIGHT = (1, 0)

        self.snake = [(100, 100), (90, 100), (80, 100)]
        self.direction = self.RIGHT
        self.food = self.generate_food()

        self.speeds = {"慢": 5, "中": 10, "快": 15}
        self.speed = self.speeds["慢"]

        self.running = True
        self.clock = pygame.time.Clock()

    def generate_food(self):
        return (random.randint(0, (self.WIDTH // self.CELL_SIZE) - 1) * self.CELL
            _SIZE,
                random.randint(0, (self.HEIGHT // self.CELL_SIZE) - 1) * self.CEL
            L_SIZE)

```

```

def handle_events(self):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            self.running = False
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP and self.direction != self.DOWN:
                self.direction = self.UP
            elif event.key == pygame.K_DOWN and self.direction != self.UP:
                self.direction = self.DOWN
            elif event.key == pygame.K_LEFT and self.direction != self.RIGHT:
                self.direction = self.LEFT
            elif event.key == pygame.K_RIGHT and self.direction != self.LEFT:
                self.direction = self.RIGHT
            elif event.key == pygame.K_1:
                self.speed = self.speeds["慢"]
            elif event.key == pygame.K_2:
                self.speed = self.speeds["中"]
            elif event.key == pygame.K_3:
                self.speed = self.speeds["快"]

def update(self):
    new_head = (self.snake[0][0] + self.direction[0] * self.CELL_SIZE,
                self.snake[0][1] + self.direction[1] * self.CELL_SIZE)

    if new_head in self.snake or new_head[0] < 0 or new_head[1] < 0 or \
        new_head[0] >= self.WIDTH or new_head[1] >= self.HEIGHT:
        self.running = False
        return

    self.snake.insert(0, new_head)

    if new_head == self.food:
        self.food = self.generate_food()
    else:
        self.snake.pop()

def draw(self):
    self.screen.fill(self.BLACK)
    for segment in self.snake:
        pygame.draw.rect(self.screen, self.GREEN, (segment[0], segment[1], self.CELL_SIZE, self.CELL_SIZE))
    pygame.draw.rect(self.screen, self.RED, (self.food[0], self.food[1], self.CELL_SIZE, self.CELL_SIZE))

    font = pygame.font.SysFont(None, 24)

```

```

        text = font.render(f"速度: {'慢' if self.speed == self.speeds['慢'] else
        '中' if self.speed == self.speeds['中'] else '快'} (按1,2,3切换)", True, self.WH
        ITE)
        self.screen.blit(text, (10, 10))

        pygame.display.flip()

    def run(self):
        while self.running:
            self.handle_events()
            self.update()
            self.draw()
            self.clock.tick(self.speed)
        pygame.quit()

if __name__ == "__main__":
    game = SnakeGame()
    game.run()

```

v1.3

我：界面没有显示中文，全是方框了，修正一下

它：我已经修正了字体问题，现在游戏界面应该可以正确显示中文了。我使用了 simhei（黑体），这是一款常见的中文字体。如果仍然有问题，可以尝试更换为其他中文字体，如 simsun（宋体）。你可以试试看效果如何！

```

import pygame
import random

class SnakeGame:
    def __init__(self):
        pygame.init()
        self.WIDTH, self.HEIGHT = 600, 400
        self.CELL_SIZE = 20
        self.screen = pygame.display.set_mode((self.WIDTH, self.HEIGHT))
        pygame.display.set_caption("贪吃蛇")

        self.WHITE = (255, 255, 255)
        self.GREEN = (0, 255, 0)
        self.RED = (255, 0, 0)
        self.BLACK = (0, 0, 0)

        self.UP = (0, -1)
        self.DOWN = (0, 1)
        self.LEFT = (-1, 0)

```

```
self.RIGHT = (1, 0)

self.snake = [(100, 100), (90, 100), (80, 100)]
self.direction = self.RIGHT
self.food = self.generate_food()

self.speeds = {"慢": 5, "中": 10, "快": 15}
self.speed = self.speeds["慢"]

self.running = True
self.clock = pygame.time.Clock()

def generate_food(self):
    return (random.randint(0, (self.WIDTH // self.CELL_SIZE) - 1) * self.CELL_SIZE,
            random.randint(0, (self.HEIGHT // self.CELL_SIZE) - 1) * self.CELL_SIZE)

def handle_events(self):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            self.running = False
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP and self.direction != self.DOWN:
                self.direction = self.UP
            elif event.key == pygame.K_DOWN and self.direction != self.UP:
                self.direction = self.DOWN
            elif event.key == pygame.K_LEFT and self.direction != self.RIGHT:
                self.direction = self.LEFT
            elif event.key == pygame.K_RIGHT and self.direction != self.LEFT:
                self.direction = self.RIGHT
            elif event.key == pygame.K_1:
                self.speed = self.speeds["慢"]
            elif event.key == pygame.K_2:
                self.speed = self.speeds["中"]
            elif event.key == pygame.K_3:
                self.speed = self.speeds["快"]

def update(self):
    new_head = (self.snake[0][0] + self.direction[0] * self.CELL_SIZE,
                self.snake[0][1] + self.direction[1] * self.CELL_SIZE)

    if new_head in self.snake or new_head[0] < 0 or new_head[1] < 0 or \
        new_head[0] >= self.WIDTH or new_head[1] >= self.HEIGHT:
        self.running = False
    return
```

```

        self.snake.insert(0, new_head)

        if new_head == self.food:
            self.food = self.generate_food()
        else:
            self.snake.pop()

    def draw(self):
        self.screen.fill(self.BLACK)
        for segment in self.snake:
            pygame.draw.rect(self.screen, self.GREEN, (segment[0], segment[1], self.CELL_SIZE, self.CELL_SIZE))
            pygame.draw.rect(self.screen, self.RED, (self.food[0], self.food[1], self.CELL_SIZE, self.CELL_SIZE))

        font = pygame.font.Font(pygame.font.match_font('simhei'), 24)
        text = font.render(f"速度: {'慢' if self.speed == self.speeds['慢'] else '中' if self.speed == self.speeds['中'] else '快'} (按1,2,3切换)", True, self.WHITE)
        self.screen.blit(text, (10, 10))

    pygame.display.flip()

    def run(self):
        while self.running:
            self.handle_events()
            self.update()
            self.draw()
            self.clock.tick(self.speed)
        pygame.quit()

if __name__ == "__main__":
    game = SnakeGame()
    game.run()

```

v1.4

达到chatgpt最大次数了，需要重新创建对话。

我：我给你一段python贪吃蛇代码，需要你帮我优化一下，回复明白，我就发代码

接着发它v1.3的代码

它：

代码看起来挺好的，但有一些小地方可以优化或改进。以下是我的优化建议：

1. 提高代码可读性：

- 将常量（如颜色和方向）定义为类外部常量，避免在每个方法中重复定义。
- 使用 `Enum` 来定义方向（`UP`，`DOWN`，`LEFT`，`RIGHT`），这样可以增加可读性和避免硬编码。

2. 优化 `update` 方法：

- 目前的 `update` 方法中，计算新的蛇头位置是通过 `+` 运算。可以考虑将其封装成一个函数，使代码更清晰。

3. 减少重复代码：

- 生成食物和检查蛇头碰撞的代码可以提取为一个函数，提高代码复用性。

4. 游戏速度切换功能：

- 当前按键事件的判断有点硬编码，速度切换可以考虑使用 `dict` 或 `list` 来简化逻辑。

我将根据上述建议对代码进行优化，并进行简化：

```
import pygame
import random
from enum import Enum

class Direction(Enum):
    UP = (0, -1)
    DOWN = (0, 1)
    LEFT = (-1, 0)
    RIGHT = (1, 0)

class SnakeGame:
    WIDTH, HEIGHT = 600, 400
    CELL_SIZE = 20
    WHITE = (255, 255, 255)
    GREEN = (0, 255, 0)
    RED = (255, 0, 0)
    BLACK = (0, 0, 0)

    SPEEDS = {"慢": 5, "中": 10, "快": 15}
```

```

def __init__(self):
    pygame.init()
    self.screen = pygame.display.set_mode((self.WIDTH, self.HEIGHT))
    pygame.display.set_caption("贪吃蛇")

    self.snake = [(100, 100), (90, 100), (80, 100)]
    self.direction = Direction.RIGHT
    self.food = self.generate_food()
    self.speed = self.SPEEDS["慢"]

    self.running = True
    self.clock = pygame.time.Clock()

    def generate_food(self):
        return (random.randint(0, (self.WIDTH // self.CELL_SIZE) - 1) * self.CELL_SIZE,
                random.randint(0, (self.HEIGHT // self.CELL_SIZE) - 1) * self.CELL_SIZE)

    def handle_events(self):
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                self.running = False
            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_UP and self.direction != Direction.DOWN:
                    self.direction = Direction.UP
                elif event.key == pygame.K_DOWN and self.direction != Direction.UP:
                    self.direction = Direction.DOWN
                elif event.key == pygame.K_LEFT and self.direction != Direction.RIGHT:
                    self.direction = Direction.LEFT
                elif event.key == pygame.K_RIGHT and self.direction != Direction.LEFT:
                    self.direction = Direction.RIGHT
                elif event.key == pygame.K_1:
                    self.speed = self.SPEEDS["慢"]
                elif event.key == pygame.K_2:
                    self.speed = self.SPEEDS["中"]
                elif event.key == pygame.K_3:
                    self.speed = self.SPEEDS["快"]

    def update(self):
        head_x, head_y = self.snake[0]
        direction_x, direction_y = self.direction.value
        new_head = (head_x + direction_x * self.CELL_SIZE, head_y + direction_y *

```

```

self.CELL_SIZE)

    if self.is_collision(new_head):
        self.running = False
        return

    self.snake.insert(0, new_head)
    if new_head == self.food:
        self.food = self.generate_food()
    else:
        self.snake.pop()

    def is_collision(self, position):
        x, y = position
        return (position in self.snake or x < 0 or y < 0 or x >= self.WIDTH or y
        >= self.HEIGHT)

    def draw(self):
        self.screen.fill(self.BLACK)
        for segment in self.snake:
            pygame.draw.rect(self.screen, self.GREEN, (segment[0], segment[1], se
lf.CELL_SIZE, self.CELL_SIZE))
            pygame.draw.rect(self.screen, self.RED, (self.food[0], self.food[1], sel
f.CELL_SIZE, self.CELL_SIZE))

        font = pygame.font.Font(pygame.font.match_font('simhei'), 24)
        speed_text = f"速度: {'慢' if self.speed == self.SPEEDS['慢'] else '中' i
f self.speed == self.SPEEDS['中'] else '快'} (按1,2,3切换)"
        text = font.render(speed_text, True, self.WHITE)
        self.screen.blit(text, (10, 10))

    pygame.display.flip()

    def run(self):
        while self.running:
            self.handle_events()
            self.update()
            self.draw()
            self.clock.tick(self.speed)
        pygame.quit()

if __name__ == "__main__":
    game = SnakeGame()
    game.run()

```

优化说明：

1. `Direction` 枚举类：通过 `Enum` 将方向变成了更加结构化的类型，避免了硬编码的方向值。
2. 碰撞检查方法：将碰撞检查封装到 `is_collision` 方法中，简化了 `update` 方法的逻辑。
3. `SPEEDS` 字典：将速度映射放到字典中，简化了代码逻辑，并方便以后扩展。
4. 游戏速度设置：利用字典快速查找速度并设置。

这样优化后，代码结构更加清晰、可维护性更强。如果有其他功能需求或优化点，欢迎继续沟通！

v1.5

我：很好，接下来优化，当碰到墙时，蛇从墙壁穿过，从另一侧出来，不要结束游戏，游戏窗口背景设置为白色。重写一下。

它：

好的，我明白您的需求。接下来我们需要修改的地方：

1. 蛇穿越墙壁：当蛇的头部碰到墙壁时，它将从另一侧重新出现，而不是结束游戏。具体来说，就是当蛇的头部移出屏幕边界时，它应该从另一边重新进入。
2. 背景色修改：将背景色设置为白色。

我会根据这些需求进行修改，以下是更新后的代码：

```
import pygame
import random
from enum import Enum

class Direction(Enum):
    UP = (0, -1)
    DOWN = (0, 1)
    LEFT = (-1, 0)
    RIGHT = (1, 0)

class SnakeGame:
    WIDTH, HEIGHT = 600, 400
    CELL_SIZE = 20
    WHITE = (255, 255, 255)
    GREEN = (0, 255, 0)
    RED = (255, 0, 0)
    BLACK = (0, 0, 0)
```

```

SPEEDS = {"慢": 5, "中": 10, "快": 15}

def __init__(self):
    pygame.init()
    self.screen = pygame.display.set_mode((self.WIDTH, self.HEIGHT))
    pygame.display.set_caption("贪吃蛇")

    self.snake = [(100, 100), (90, 100), (80, 100)]
    self.direction = Direction.RIGHT
    self.food = self.generate_food()
    self.speed = self.SPEEDS["慢"]

    self.running = True
    self.clock = pygame.time.Clock()

def generate_food(self):
    return (random.randint(0, (self.WIDTH // self.CELL_SIZE) - 1) * self.CELL_SIZE,
            random.randint(0, (self.HEIGHT // self.CELL_SIZE) - 1) * self.CELL_SIZE)

def handle_events(self):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            self.running = False
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP and self.direction != Direction.DOWN:
                self.direction = Direction.UP
            elif event.key == pygame.K_DOWN and self.direction != Direction.UP:
                self.direction = Direction.DOWN
            elif event.key == pygame.K_LEFT and self.direction != Direction.RIGHT:
                self.direction = Direction.LEFT
            elif event.key == pygame.K_RIGHT and self.direction != Direction.LEFT:
                self.direction = Direction.RIGHT
            elif event.key == pygame.K_1:
                self.speed = self.SPEEDS["慢"]
            elif event.key == pygame.K_2:
                self.speed = self.SPEEDS["中"]
            elif event.key == pygame.K_3:
                self.speed = self.SPEEDS["快"]

def update(self):
    head_x, head_y = self.snake[0]

```

```

        direction_x, direction_y = self.direction.value
        new_head = (head_x + direction_x * self.CELL_SIZE, head_y + direction_y *
self.CELL_SIZE)

        # Check if the snake hits the wall and wrap around
        new_head = self.wrap_around(new_head)

        if new_head in self.snake:
            self.running = False
            return

        self.snake.insert(0, new_head)
        if new_head == self.food:
            self.food = self.generate_food()
        else:
            self.snake.pop()

    def wrap_around(self, position):
        x, y = position
        # Wrap around the screen edges
        if x < 0:
            x = self.WIDTH - self.CELL_SIZE
        elif x >= self.WIDTH:
            x = 0
        if y < 0:
            y = self.HEIGHT - self.CELL_SIZE
        elif y >= self.HEIGHT:
            y = 0
        return (x, y)

    def draw(self):
        self.screen.fill(self.WHITE) # Set background to white
        for segment in self.snake:
            pygame.draw.rect(self.screen, self.GREEN, (segment[0], segment[1], se
lf.CELL_SIZE, self.CELL_SIZE))
            pygame.draw.rect(self.screen, self.RED, (self.food[0], self.food[1], sel
f.CELL_SIZE, self.CELL_SIZE))

        font = pygame.font.Font(pygame.font.match_font('simhei'), 24)
        speed_text = f"速度: {'慢' if self.speed == self.SPEEDS['慢'] else '中' i
f self.speed == self.SPEEDS['中'] else '快'} (按1,2,3切换)"
        text = font.render(speed_text, True, self.BLACK) # Text color set to bla
ck for better contrast
        self.screen.blit(text, (10, 10))

    pygame.display.flip()

```

```

def run(self):
    while self.running:
        self.handle_events()
        self.update()
        self.draw()
        self.clock.tick(self.speed)
    pygame.quit()

if __name__ == "__main__":
    game = SnakeGame()
    game.run()

```

更新说明：

1. 蛇穿越墙壁：新增了 `wrap_around` 方法，处理蛇头穿越屏幕边界的情况。当蛇头越过屏幕边界时，会从相反的边缘重新出现。`update` 方法中调用了 `wrap_around` 来处理边界碰撞。
2. 背景色更改：`draw` 方法中将背景色修改为白色，使用 `self.screen.fill(self.WHITE)` 来填充背景。
3. 游戏结束判定：如果蛇头与自身发生碰撞，游戏会结束。

通过这些更新，蛇现在可以穿越屏幕边界，并从另一侧出来，而不再结束游戏。背景也变为白色，界面更加明亮。如果还有其他需求或问题，随时告诉我！

v1.6

我：设置食物为圆形

它：

好的，您想将食物的形状从矩形更改为圆形。我们可以通过修改 `draw` 方法来实现这一点，使用 `pygame.draw.circle` 来绘制食物，而不是使用 `pygame.draw.rect`。

以下是更新后的代码：

```

import pygame
import random
from enum import Enum

class Direction(Enum):
    UP = (0, -1)
    DOWN = (0, 1)

```

```

LEFT = (-1, 0)
RIGHT = (1, 0)

class SnakeGame:
    WIDTH, HEIGHT = 600, 400
    CELL_SIZE = 20
    WHITE = (255, 255, 255)
    GREEN = (0, 255, 0)
    RED = (255, 0, 0)
    BLACK = (0, 0, 0)

    SPEEDS = {"慢": 5, "中": 10, "快": 15}

    def __init__(self):
        pygame.init()
        self.screen = pygame.display.set_mode((self.WIDTH, self.HEIGHT))
        pygame.display.set_caption("贪吃蛇")

        self.snake = [(100, 100), (90, 100), (80, 100)]
        self.direction = Direction.RIGHT
        self.food = self.generate_food()
        self.speed = self.SPEEDS["慢"]

        self.running = True
        self.clock = pygame.time.Clock()

    def generate_food(self):
        return (random.randint(0, (self.WIDTH // self.CELL_SIZE) - 1) * self.CELL_SIZE,
                random.randint(0, (self.HEIGHT // self.CELL_SIZE) - 1) * self.CELL_SIZE)

    def handle_events(self):
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                self.running = False
            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_UP and self.direction != Direction.DOWN:
                    self.direction = Direction.UP
                elif event.key == pygame.K_DOWN and self.direction != Direction.UP:
                    self.direction = Direction.DOWN
                elif event.key == pygame.K_LEFT and self.direction != Direction.RIGHT:
                    self.direction = Direction.LEFT

```

```

        elif event.key == pygame.K_RIGHT and self.direction != Direction.
LEFT:
        self.direction = Direction.RIGHT
        elif event.key == pygame.K_1:
            self.speed = self.SPEEDS["慢"]
        elif event.key == pygame.K_2:
            self.speed = self.SPEEDS["中"]
        elif event.key == pygame.K_3:
            self.speed = self.SPEEDS["快"]

    def update(self):
        head_x, head_y = self.snake[0]
        direction_x, direction_y = self.direction.value
        new_head = (head_x + direction_x * self.CELL_SIZE, head_y + direction_y *
self.CELL_SIZE)

        # Check if the snake hits the wall and wrap around
        new_head = self.wrap_around(new_head)

        if new_head in self.snake:
            self.running = False
            return

        self.snake.insert(0, new_head)
        if new_head == self.food:
            self.food = self.generate_food()
        else:
            self.snake.pop()

    def wrap_around(self, position):
        x, y = position
        # Wrap around the screen edges
        if x < 0:
            x = self.WIDTH - self.CELL_SIZE
        elif x >= self.WIDTH:
            x = 0
        if y < 0:
            y = self.HEIGHT - self.CELL_SIZE
        elif y >= self.HEIGHT:
            y = 0
        return (x, y)

    def draw(self):
        self.screen.fill(self.WHITE) # Set background to white
        for segment in self.snake:
            pygame.draw.rect(self.screen, self.GREEN, (segment[0], segment[1], se

```

```
lf.CELL_SIZE, self.CELL_SIZE))

    # Draw food as a circle
    food_radius = self.CELL_SIZE // 2
    pygame.draw.circle(self.screen, self.RED, (self.food[0] + food_radius, se
lf.food[1] + food_radius), food_radius)

    font = pygame.font.Font(pygame.font.match_font('simhei'), 24)
    speed_text = f"速度: {'慢' if self.speed == self.SPEEDS['慢'] else '中' i
f self.speed == self.SPEEDS['中'] else '快'} (按1,2,3切换)"
    text = font.render(speed_text, True, self.BLACK) # Text color set to bla
ck for better contrast
    self.screen.blit(text, (10, 10))

    pygame.display.flip()

def run(self):
    while self.running:
        self.handle_events()
        self.update()
        self.draw()
        self.clock.tick(self.speed)
    pygame.quit()

if __name__ == "__main__":
    game = SnakeGame()
    game.run()
```

主要变化：

1. 食物绘制：在 `draw` 方法中，我使用了 `pygame.draw.circle` 来绘制食物，而不是矩形。食物的圆心位置是食物的 `(x, y)` 位置，加上半径（即 `CELL_SIZE // 2`），使得它居中显示。

其他说明：

- 圆形食物的大小：食物的半径是 `CELL_SIZE // 2`，这样它会适应当前设置的单元格大小。
- 食物颜色：食物的颜色保持为红色（`RED`）。

现在游戏中的食物已经是圆形的了。如果您有更多要求或想进一步调整，欢迎继续交流！

v1.7

由于新开对话，所以使用v1.6版本代码让它优化。

我：我给你一段python贪吃蛇代码，需要你帮我优化一下，回复明白，我就发代码
它回复了很多，不用理它。

我：很好。需要你帮我添加积分功能，默认每个圆形食物加1分，随机每3~5个圆形食物后生成一个彩色五角星食物，吃了它加5分。注意吃了彩色五角星食物后，需要将连续吃的圆形食物数重置为0。并且在界面右上角显示吃了多少个食物了

它：

好的，我会按照您的要求添加积分功能，并且根据说明生成普通圆形食物和特殊彩色五角星食物。每次吃掉普通食物时加 1 分，吃掉五角星食物加 5 分，并且每吃掉五角星食物后会重置连续吃的圆形食物数。

以下是更新后的代码：

```
import pygame
import random
from enum import Enum

class Direction(Enum):
    UP = (0, -1)
    DOWN = (0, 1)
    LEFT = (-1, 0)
    RIGHT = (1, 0)

class SnakeGame:
    WIDTH, HEIGHT = 600, 400
    CELL_SIZE = 20
    WHITE = (255, 255, 255)
    GREEN = (0, 255, 0)
    RED = (255, 0, 0)
    BLACK = (0, 0, 0)
    COLORS = [(255, 215, 0), (255, 0, 255), (0, 255, 255), (255, 69, 0), (0, 255, 0)]

    SPEEDS = {"慢": 5, "中": 10, "快": 15}

    def __init__(self):
        pygame.init()
        self.screen = pygame.display.set_mode((self.WIDTH, self.HEIGHT))
        pygame.display.set_caption("贪吃蛇")

        self.snake = [(100, 100), (90, 100), (80, 100)]
        self.direction = Direction.RIGHT
        self.food = self.generate_food()
```

```
self.speed = self.SPEEDS["慢"]

self.running = True
self.clock = pygame.time.Clock()
self.score = 0
self.continuous_regular_food = 0 # 记录连续吃的普通食物数

def generate_food(self):
    """生成随机食物的位置（可能是普通圆形食物，也可能是五角星食物）"""
    if self.continuous_regular_food >= random.randint(3, 5):
        # 生成五角星食物
        food_type = 'star'
        food_pos = (random.randint(0, (self.WIDTH // self.CELL_SIZE) - 1) * s
elf.CELL_SIZE,
                    random.randint(0, (self.HEIGHT // self.CELL_SIZE) - 1) *
self.CELL_SIZE)
    else:
        # 生成普通圆形食物
        food_type = 'circle'
        food_pos = (random.randint(0, (self.WIDTH // self.CELL_SIZE) - 1) * s
elf.CELL_SIZE,
                    random.randint(0, (self.HEIGHT // self.CELL_SIZE) - 1) *
self.CELL_SIZE)
    return food_pos, food_type

def handle_events(self):
    """处理用户输入的事件"""
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            self.running = False
        elif event.type == pygame.KEYDOWN:
            self.change_direction(event)
            self.change_speed(event)

def change_direction(self, event):
    """根据按键改变方向"""
    if event.key == pygame.K_UP and self.direction != Direction.DOWN:
        self.direction = Direction.UP
    elif event.key == pygame.K_DOWN and self.direction != Direction.UP:
        self.direction = Direction.DOWN
    elif event.key == pygame.K_LEFT and self.direction != Direction.RIGHT:
        self.direction = Direction.LEFT
    elif event.key == pygame.K_RIGHT and self.direction != Direction.LEFT:
        self.direction = Direction.RIGHT

def change_speed(self, event):
```

```

    """根据按键改变速度"""
    if event.key == pygame.K_1:
        self.speed = self.SPEEDS["慢"]
    elif event.key == pygame.K_2:
        self.speed = self.SPEEDS["中"]
    elif event.key == pygame.K_3:
        self.speed = self.SPEEDS["快"]

    def update(self):
        """更新游戏状态"""
        head_x, head_y = self.snake[0]
        direction_x, direction_y = self.direction.value
        new_head = (head_x + direction_x * self.CELL_SIZE, head_y + direction_y *
self.CELL_SIZE)

        # 检查是否撞到墙壁并实现循环
        new_head = self.wrap_around(new_head)

        if new_head in self.snake:
            self.running = False
            return

        self.snake.insert(0, new_head)

        # 检查是否吃到了食物
        food_position, food_type = self.food
        if new_head == food_position:
            if food_type == 'circle':
                self.score += 1 # 吃掉普通圆形食物加1分
                self.continuous_regular_food += 1 # 连续吃的普通食物数加1
            elif food_type == 'star':
                self.score += 5 # 吃掉五角星食物加5分
                self.continuous_regular_food = 0 # 吃了五角星后重置普通食物数
            self.food = self.generate_food() # 生成新的食物
        else:
            self.snake.pop()

    def wrap_around(self, position):
        """检查并处理蛇是否超出边界并且让其循环"""
        x, y = position
        x %= self.WIDTH
        y %= self.HEIGHT
        return (x, y)

    def draw_star(self, center, size):
        """绘制五角星食物"""

```

```
        star_points = [  
            (center[0], center[1] - size),  
            (center[0] + size * 0.2, center[1] - size * 0.3),  
            (center[0] + size, center[1] - size * 0.3),  
            (center[0] + size * 0.3, center[1]),  
            (center[0] + size * 0.5, center[1] + size),  
            (center[0], center[1] + size * 0.4),  
            (center[0] - size * 0.5, center[1] + size),  
            (center[0] - size * 0.3, center[1]),  
            (center[0] - size, center[1] - size * 0.3),  
            (center[0] - size * 0.2, center[1] - size * 0.3)  
        ]  
        pygame.draw.polygon(self.screen, random.choice(self.COLORS), star_points)  
  
    def draw(self):  
        """绘制游戏画面"""  
        self.screen.fill(self.WHITE) # 设置背景为白色  
        for segment in self.snake:  
            pygame.draw.rect(self.screen, self.GREEN, (segment[0], segment[1], se  
lf.CELL_SIZE, self.CELL_SIZE))  
  
        # 绘制食物  
        food_position, food_type = self.food  
        if food_type == 'circle':  
            food_radius = self.CELL_SIZE // 2  
            pygame.draw.circle(self.screen, self.RED, (food_position[0] + food_ra  
dius, food_position[1] + food_radius), food_radius)  
        elif food_type == 'star':  
            self.draw_star(food_position, self.CELL_SIZE)  
  
        # 显示积分和吃掉的食物数量  
        font = pygame.font.Font(pygame.font.match_font('simhei'), 24)  
        score_text = f"得分: {self.score} | 吃了 {self.score} 个食物"  
        text = font.render(score_text, True, self.BLACK)  
        self.screen.blit(text, (self.WIDTH - 250, 10))  
  
        pygame.display.flip()  
  
    def run(self):  
        """运行游戏循环"""  
        while self.running:  
            self.handle_events()  
            self.update()  
            self.draw()  
            self.clock.tick(self.speed)  
        pygame.quit()
```

```
if __name__ == "__main__":
    game = SnakeGame()
    game.run()
```

主要改动：

1. 积分系统：每吃一个普通食物得 1 分，每吃一个五角星食物得 5 分。
2. 五角星食物：随机生成一个五角星食物，当连续吃了 3 到 5 个普通食物后，才会生成五角星食物，并重置连续吃的普通食物数。
3. 显示积分：在右上角显示当前得分和吃掉的食物数量。
4. 五角星绘制：增加了一个方法 `draw_star` 来绘制五角星食物，使用随机颜色。

如果您有任何进一步的需求或调整，欢迎告诉我！

生成的代码有一点问题，把 `self.continuous_regular_food = 0` `*** 记录连续吃的普通食物数*` 放到 `pygame.display.set_caption("贪吃蛇")` 之后就好了。

v1.8

我：有几个问题，得分和吃了食物个数不一样。修正一下

它：明白，您的需求是 得分和吃掉的食物数量分开计算，而不是直接使用得分作为吃掉的食物个数。我会修正代码，确保 得分 正确计算（普通食物+1，五角星+5），同时 吃掉的食物总数 也正确累加（每次吃食物都加1）。

更新后的代码如下：

```
import pygame
import random
from enum import Enum

class Direction(Enum):
    UP = (0, -1)
    DOWN = (0, 1)
    LEFT = (-1, 0)
    RIGHT = (1, 0)

class SnakeGame:
    WIDTH, HEIGHT = 600, 400
    CELL_SIZE = 20
    WHITE = (255, 255, 255)
    GREEN = (0, 255, 0)
```

```

RED = (255, 0, 0)
BLACK = (0, 0, 0)
COLORS = [(255, 215, 0), (255, 0, 255), (0, 255, 255), (255, 69, 0), (0, 255,
0)]

SPEEDS = {"慢": 5, "中": 10, "快": 15}

def __init__(self):
    pygame.init()
    self.screen = pygame.display.set_mode((self.WIDTH, self.HEIGHT))
    pygame.display.set_caption("贪吃蛇")

    self.snake = [(100, 100), (90, 100), (80, 100)]
    self.direction = Direction.RIGHT
    self.food = self.generate_food()
    self.speed = self.SPEEDS["慢"]

    self.running = True
    self.clock = pygame.time.Clock()
    self.score = 0 # 记录得分
    self.food_count = 0 # 记录吃掉的食物总数
    self.continuous_regular_food = 0 # 记录连续吃的普通食物数

def generate_food(self):
    """生成随机食物的位置（可能是普通圆形食物，也可能是五角星食物）"""
    if self.continuous_regular_food >= random.randint(3, 5):
        # 生成五角星食物
        food_type = 'star'
        food_pos = (random.randint(0, (self.WIDTH // self.CELL_SIZE) - 1) * s
elf.CELL_SIZE,
                    random.randint(0, (self.HEIGHT // self.CELL_SIZE) - 1) *
self.CELL_SIZE)
    else:
        # 生成普通圆形食物
        food_type = 'circle'
        food_pos = (random.randint(0, (self.WIDTH // self.CELL_SIZE) - 1) * s
elf.CELL_SIZE,
                    random.randint(0, (self.HEIGHT // self.CELL_SIZE) - 1) *
self.CELL_SIZE)
    return food_pos, food_type

def handle_events(self):
    """处理用户输入的事件"""
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            self.running = False

```

```
        elif event.type == pygame.KEYDOWN:
            self.change_direction(event)
            self.change_speed(event)

    def change_direction(self, event):
        """根据按键改变方向"""
        if event.key == pygame.K_UP and self.direction != Direction.DOWN:
            self.direction = Direction.UP
        elif event.key == pygame.K_DOWN and self.direction != Direction.UP:
            self.direction = Direction.DOWN
        elif event.key == pygame.K_LEFT and self.direction != Direction.RIGHT:
            self.direction = Direction.LEFT
        elif event.key == pygame.K_RIGHT and self.direction != Direction.LEFT:
            self.direction = Direction.RIGHT

    def change_speed(self, event):
        """根据按键改变速度"""
        if event.key == pygame.K_1:
            self.speed = self.SPEEDS["慢"]
        elif event.key == pygame.K_2:
            self.speed = self.SPEEDS["中"]
        elif event.key == pygame.K_3:
            self.speed = self.SPEEDS["快"]

    def update(self):
        """更新游戏状态"""
        head_x, head_y = self.snake[0]
        direction_x, direction_y = self.direction.value
        new_head = (head_x + direction_x * self.CELL_SIZE, head_y + direction_y *
self.CELL_SIZE)

        # 检查是否撞到墙壁并实现循环
        new_head = self.wrap_around(new_head)

        if new_head in self.snake:
            self.running = False
            return

        self.snake.insert(0, new_head)

        # 检查是否吃到了食物
        food_position, food_type = self.food
        if new_head == food_position:
            self.food_count += 1 # 每次吃到食物, 食物计数+1
            if food_type == 'circle':
                self.score += 1 # 普通食物+1分
```

```

        self.continuous_regular_food += 1 # 记录连续普通食物次数
    elif food_type == 'star':
        self.score += 5 # 五角星食物+5分
        self.continuous_regular_food = 0 # 吃五角星后重置普通食物计数
        self.food = self.generate_food() # 生成新的食物
    else:
        self.snake.pop()

def wrap_around(self, position):
    """检查并处理蛇是否超出边界并且让其循环"""
    x, y = position
    x %= self.WIDTH
    y %= self.HEIGHT
    return (x, y)

def draw_star(self, center, size):
    """绘制五角星食物"""
    star_points = [
        (center[0], center[1] - size),
        (center[0] + size * 0.2, center[1] - size * 0.3),
        (center[0] + size, center[1] - size * 0.3),
        (center[0] + size * 0.3, center[1]),
        (center[0] + size * 0.5, center[1] + size),
        (center[0], center[1] + size * 0.4),
        (center[0] - size * 0.5, center[1] + size),
        (center[0] - size * 0.3, center[1]),
        (center[0] - size, center[1] - size * 0.3),
        (center[0] - size * 0.2, center[1] - size * 0.3)
    ]
    pygame.draw.polygon(self.screen, random.choice(self.COLORS), star_points)

def draw(self):
    """绘制游戏画面"""
    self.screen.fill(self.WHITE) # 设置背景为白色
    for segment in self.snake:
        pygame.draw.rect(self.screen, self.GREEN, (segment[0], segment[1], self.CELL_SIZE, self.CELL_SIZE))

    # 绘制食物
    food_position, food_type = self.food
    if food_type == 'circle':
        food_radius = self.CELL_SIZE // 2
        pygame.draw.circle(self.screen, self.RED, (food_position[0] + food_radius, food_position[1] + food_radius), food_radius)
    elif food_type == 'star':
        self.draw_star(food_position, self.CELL_SIZE)

```

```

        # 显示得分和吃掉的食物数量
        font = pygame.font.Font(pygame.font.match_font('simhei'), 24)
        score_text = f"得分: {self.score} | 吃了 {self.food_count} 个食物"
        text = font.render(score_text, True, self.BLACK)
        self.screen.blit(text, (self.WIDTH - 250, 10))

    pygame.display.flip()

    def run(self):
        """运行游戏循环"""
        while self.running:
            self.handle_events()
            self.update()
            self.draw()
            self.clock.tick(self.speed)
        pygame.quit()

if __name__ == "__main__":
    game = SnakeGame()
    game.run()

```

修正内容：

1. 单独增加 `self.food_count` 变量，用于记录吃掉的食物总数，每吃一个食物都会+1。
2. 得分 `self.score` 计算不变，普通食物+1，五角星食物+5。
3. 界面右上角正确显示 `得分: X | 吃了 Y 个食物`，保证数据正确。

这样 `得分` 和 `吃掉的食物总数` 就是两个不同的统计了。您可以运行看看，确认是否符合您的要求！

v1.9

我：很好。五角星是很大的加分项，要给它设置过期时间，但是过期时间需要和蛇的位置有关。你再修正一下。

它：

好的，我会给五角星食物设置过期时间，但这个时间要和蛇的位置有关，确保蛇不能一直拖延时间来等五角星消失。

优化方案

1. 五角星食物的过期时间：

- 设定一个 `star_lifetime` 变量，记录五角星的剩余寿命（帧数）。

- 当生成五角星时，计算它到蛇头的曼哈顿距离（距离 = $dx + dy$ ）。
- 过期时间 = 距离 × 一个系数，让远的五角星存在更久，近的五角星存在时间短。
- 每帧减少 `star_lifetime`，如果到 0，五角星就消失，重新生成普通食物。

2. 蛇吃普通食物后，重新评估是否要生成五角星，防止连续出现五角星。

修正后代码

```
import pygame
import random
from enum import Enum

class Direction(Enum):
    UP = (0, -1)
    DOWN = (0, 1)
    LEFT = (-1, 0)
    RIGHT = (1, 0)

class SnakeGame:
    WIDTH, HEIGHT = 600, 400
    CELL_SIZE = 20
    WHITE = (255, 255, 255)
    GREEN = (0, 255, 0)
    RED = (255, 0, 0)
    BLACK = (0, 0, 0)
    COLORS = [(255, 215, 0), (255, 0, 255), (0, 255, 255), (255, 69, 0), (0, 255, 0)]

    SPEEDS = {"慢": 5, "中": 10, "快": 15}

    def __init__(self):
        pygame.init()
        self.screen = pygame.display.set_mode((self.WIDTH, self.HEIGHT))
        pygame.display.set_caption("贪吃蛇")

        self.snake = [(100, 100), (90, 100), (80, 100)]
        self.direction = Direction.RIGHT
        self.food = self.generate_food()
        self.speed = self.SPEEDS["慢"]

        self.running = True
        self.clock = pygame.time.Clock()
```

```

self.score = 0 # 记录得分
self.food_count = 0 # 记录吃掉的食物总数
self.continuous_regular_food = 0 # 记录连续吃的普通食物数
self.star_lifetime = 0 # 五角星的剩余寿命

def generate_food(self):
    """生成食物，可能是普通圆形食物或五角星食物"""
    if self.continuous_regular_food >= random.randint(3, 5):
        food_type = 'star'
        food_pos = self.get_random_position()

        # 计算蛇头到食物的曼哈顿距离
        head_x, head_y = self.snake[0]
        food_x, food_y = food_pos
        distance = abs(food_x - head_x) // self.CELL_SIZE + abs(food_y - head_y) // self.CELL_SIZE

        # 设置五角星的存活时间（距离 × 8 帧，最少 40 帧）
        self.star_lifetime = max(distance * 8, 40)
    else:
        food_type = 'circle'
        food_pos = self.get_random_position()

    return food_pos, food_type

def get_random_position(self):
    """获取随机位置"""
    return (random.randint(0, (self.WIDTH // self.CELL_SIZE) - 1) * self.CELL_SIZE,
            random.randint(0, (self.HEIGHT // self.CELL_SIZE) - 1) * self.CELL_SIZE)

def handle_events(self):
    """处理用户输入的事件"""
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            self.running = False
        elif event.type == pygame.KEYDOWN:
            self.change_direction(event)
            self.change_speed(event)

def change_direction(self, event):
    """根据按键改变方向"""
    if event.key == pygame.K_UP and self.direction != Direction.DOWN:
        self.direction = Direction.UP
    elif event.key == pygame.K_DOWN and self.direction != Direction.UP:

```

```
        self.direction = Direction.DOWN
    elif event.key == pygame.K_LEFT and self.direction != Direction.RIGHT:
        self.direction = Direction.LEFT
    elif event.key == pygame.K_RIGHT and self.direction != Direction.LEFT:
        self.direction = Direction.RIGHT

    def change_speed(self, event):
        """根据按键改变速度"""
        if event.key == pygame.K_1:
            self.speed = self.SPEEDS["慢"]
        elif event.key == pygame.K_2:
            self.speed = self.SPEEDS["中"]
        elif event.key == pygame.K_3:
            self.speed = self.SPEEDS["快"]

    def update(self):
        """更新游戏状态"""
        head_x, head_y = self.snake[0]
        direction_x, direction_y = self.direction.value
        new_head = (head_x + direction_x * self.CELL_SIZE, head_y + direction_y *
self.CELL_SIZE)

        # 检查是否撞到墙壁并实现循环
        new_head = self.wrap_around(new_head)

        if new_head in self.snake:
            self.running = False
            return

        self.snake.insert(0, new_head)

        # 检查是否吃到了食物
        food_position, food_type = self.food
        if new_head == food_position:
            self.food_count += 1 # 每次吃到食物, 食物计数+1
            if food_type == 'circle':
                self.score += 1
                self.continuous_regular_food += 1
            elif food_type == 'star':
                self.score += 5
                self.continuous_regular_food = 0
                self.star_lifetime = 0 # 吃掉五角星后立即过期

        self.food = self.generate_food() # 生成新食物
    else:
        self.snake.pop()
```

```

        # 处理五角星过期逻辑
        if self.food[1] == 'star' and self.star_lifetime > 0:
            self.star_lifetime -= 1
            if self.star_lifetime == 0:
                self.food = self.generate_food() # 过期后重新生成普通食物

    def wrap_around(self, position):
        """检查并处理蛇是否超出边界并且让其循环"""
        x, y = position
        x %= self.WIDTH
        y %= self.HEIGHT
        return (x, y)

    def draw_star(self, center, size):
        """绘制五角星食物"""
        star_points = [
            (center[0], center[1] - size),
            (center[0] + size * 0.2, center[1] - size * 0.3),
            (center[0] + size, center[1] - size * 0.3),
            (center[0] + size * 0.3, center[1]),
            (center[0] + size * 0.5, center[1] + size),
            (center[0], center[1] + size * 0.4),
            (center[0] - size * 0.5, center[1] + size),
            (center[0] - size * 0.3, center[1]),
            (center[0] - size, center[1] - size * 0.3),
            (center[0] - size * 0.2, center[1] - size * 0.3)
        ]
        pygame.draw.polygon(self.screen, random.choice(self.COLORS), star_points)

    def draw(self):
        """绘制游戏画面"""
        self.screen.fill(self.WHITE)
        for segment in self.snake:
            pygame.draw.rect(self.screen, self.GREEN, (segment[0], segment[1], self.CELL_SIZE, self.CELL_SIZE))

        food_position, food_type = self.food
        if food_type == 'circle':
            pygame.draw.circle(self.screen, self.RED, (food_position[0] + 10, food_position[1] + 10), 10)
        else:
            self.draw_star(food_position, self.CELL_SIZE)

        font = pygame.font.Font(None, 24)
        text = font.render(f"得分: {self.score} | 食物: {self.food_count} | 五角

```

```

星剩余时间: {self.star_lifetime}", True, self.BLACK)
        self.screen.blit(text, (self.WIDTH - 300, 10))

pygame.display.flip()

if __name__ == "__main__":
    game = SnakeGame()
    game.run()

```

这个版本保证五角星食物的过期时间随蛇的位置动态调整，让游戏更有策略性。你可以试试！

这次生成的代码没有game.run()了，再次生成时中文变方框了，一步步让它改下就好。

v1.10

我：我给你一段代码，你先理解，回复明白，然后我提出新的需求。

五角星要在一定时间后自动消失，这个时间需要和蛇的距离有关。

它：

好的，这是根据你要求修改后的完整版代码，确保星形食物能够按照生命周期消失：

```

import pygame
import random
from enum import Enum

class Direction(Enum):
    UP = (0, -1)
    DOWN = (0, 1)
    LEFT = (-1, 0)
    RIGHT = (1, 0)

class SnakeGame:
    WIDTH, HEIGHT = 600, 400
    CELL_SIZE = 20
    WHITE = (255, 255, 255)
    GREEN = (0, 255, 0)
    RED = (255, 0, 0)
    BLACK = (0, 0, 0)
    COLORS = [(255, 215, 0), (255, 0, 255), (0, 255, 255), (255, 69, 0), (0, 255, 0)]

    SPEEDS = {"慢": 5, "中": 10, "快": 15}

    def __init__(self):

```

```
self.continuous_regular_food = 0
self.snake = [(100, 100), (90, 100), (80, 100)]
self.direction = Direction.RIGHT
self.food = self.generate_food()
self.speed = self.SPEEDS["慢"]
self.running = True
self.score = 0
self.food_count = 0
self.star_lifetime = 0

pygame.init()
self.screen = pygame.display.set_mode((self.WIDTH, self.HEIGHT))
pygame.display.set_caption("贪吃蛇")
self.clock = pygame.time.Clock()

def generate_food(self):
    if self.continuous_regular_food >= random.randint(3, 5):
        food_type = 'star'
        food_pos = self.get_random_position()
        head_x, head_y = self.snake[0]
        food_x, food_y = food_pos
        distance = abs(food_x - head_x) // self.CELL_SIZE + abs(food_y - head
_y) // self.CELL_SIZE
        self.star_lifetime = max(distance * 8, 40)
    else:
        food_type = 'circle'
        food_pos = self.get_random_position()
    return food_pos, food_type

def get_random_position(self):
    return (random.randint(0, (self.WIDTH // self.CELL_SIZE) - 1) * self.CELL
_SIZE,
            random.randint(0, (self.HEIGHT // self.CELL_SIZE) - 1) * self.CEL
L_SIZE)

def handle_events(self):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            self.running = False
        elif event.type == pygame.KEYDOWN:
            self.change_direction(event)
            self.change_speed(event)

def change_direction(self, event):
    if event.key == pygame.K_UP and self.direction != Direction.DOWN:
        self.direction = Direction.UP
```

```
elif event.key == pygame.K_DOWN and self.direction != Direction.UP:
    self.direction = Direction.DOWN
elif event.key == pygame.K_LEFT and self.direction != Direction.RIGHT:
    self.direction = Direction.LEFT
elif event.key == pygame.K_RIGHT and self.direction != Direction.LEFT:
    self.direction = Direction.RIGHT

def change_speed(self, event):
    if event.key == pygame.K_1:
        self.speed = self.SPEEDS["慢"]
    elif event.key == pygame.K_2:
        self.speed = self.SPEEDS["中"]
    elif event.key == pygame.K_3:
        self.speed = self.SPEEDS["快"]

def update(self):
    head_x, head_y = self.snake[0]
    direction_x, direction_y = self.direction.value
    new_head = (head_x + direction_x * self.CELL_SIZE, head_y + direction_y *
self.CELL_SIZE)
    new_head = self.wrap_around(new_head)
    if new_head in self.snake:
        self.running = False
        return
    self.snake.insert(0, new_head)
    food_position, food_type = self.food
    if new_head == food_position:
        self.food_count += 1
        if food_type == 'circle':
            self.score += 1
            self.continuous_regular_food += 1
        elif food_type == 'star':
            self.score += 5
            self.continuous_regular_food = 0
            self.star_lifetime = 0
        self.food = self.generate_food()
    else:
        self.snake.pop()

    # 处理星形食物的生命周期
    if self.food[1] == 'star' and self.star_lifetime > 0:
        self.star_lifetime -= 1
        if self.star_lifetime == 0:
            self.food = self.generate_food()

def wrap_around(self, position):
```

```
x, y = position
x %= self.WIDTH
y %= self.HEIGHT
return (x, y)

def draw_star(self, center, size):
    star_points = [
        (center[0], center[1] - size),
        (center[0] + size * 0.2, center[1] - size * 0.3),
        (center[0] + size, center[1] - size * 0.3),
        (center[0] + size * 0.3, center[1]),
        (center[0] + size * 0.5, center[1] + size),
        (center[0], center[1] + size * 0.4),
        (center[0] - size * 0.5, center[1] + size),
        (center[0] - size * 0.3, center[1]),
        (center[0] - size, center[1] - size * 0.3),
        (center[0] - size * 0.2, center[1] - size * 0.3)
    ]
    pygame.draw.polygon(self.screen, random.choice(self.COLORS), star_points)

def draw(self):
    self.screen.fill(self.WHITE)
    for segment in self.snake:
        pygame.draw.rect(self.screen, self.GREEN, (segment[0], segment[1], self.CELL_SIZE, self.CELL_SIZE))
    food_position, food_type = self.food
    if food_type == 'circle':
        pygame.draw.circle(self.screen, self.RED, (food_position[0] + 10, food_position[1] + 10), 10)
    else:
        self.draw_star(food_position, self.CELL_SIZE)
    font = pygame.font.Font(pygame.font.match_font('simhei'), 24)
    text = font.render(f"得分: {self.score} | 食物: {self.food_count} | 五角星剩余时间: {self.star_lifetime}", True, self.BLACK)
    self.screen.blit(text, (self.WIDTH - 300, 10))
    pygame.display.flip()

def run(self):
    while self.running:
        self.handle_events()
        self.update()
        self.draw()
        self.clock.tick(self.speed)
    pygame.quit()
```

```
if __name__ == "__main__":
    game = SnakeGame()
    game.run()
```

主要修改点：

1. 生命周期处理：在 `update()` 函数中，确保星形食物的生命周期 (`star_lifetime`) 会递减，并在生命周期结束后重新生成新的食物。
2. 重新生成食物：当星形食物的生命周期结束时，会通过 `self.food = self.generate_food()` 重新生成新的食物。

现在，星形食物应该会按预期在一段时间后消失，并重新生成新的食物。你可以试试这个版本，看看是否解决了问题。

星形食物会在一定时间后消失，但再次生成时还是星形食物。

v1.11

我：很好，星形食物会在一定时间后消失，但再次生成时还是星形食物，这个要改，星形食物是加分项，没有吃到就没有了，等下次再生成。

它：

明白了，你希望在星形食物被吃掉后，不再立即生成新的星形食物，而是等到下次生成食物时才有可能再次出现星形食物。这样的话，我们需要调整星形食物的生成逻辑，确保星形食物只有在吃掉后才会再出现，而不是立即生成。

主要修改点：

1. 在星形食物被吃掉后，需要设置一个标志位，告诉系统当前没有星形食物，直到下一次生成时才有可能生成星形食物。
2. 修改 `generate_food()` 逻辑，确保星形食物不会重复生成。

修改后的代码：

```
import pygame
import random
from enum import Enum

class Direction(Enum):
    UP = (0, -1)
    DOWN = (0, 1)
    LEFT = (-1, 0)
    RIGHT = (1, 0)
```

```

class SnakeGame:
    WIDTH, HEIGHT = 600, 400
    CELL_SIZE = 20
    WHITE = (255, 255, 255)
    GREEN = (0, 255, 0)
    RED = (255, 0, 0)
    BLACK = (0, 0, 0)
    COLORS = [(255, 215, 0), (255, 0, 255), (0, 255, 255), (255, 69, 0), (0, 255,
0)]

    SPEEDS = {"慢": 5, "中": 10, "快": 15}

    def __init__(self):
        self.continuous_regular_food = 0
        self.snake = [(100, 100), (90, 100), (80, 100)]
        self.direction = Direction.RIGHT
        self.food = self.generate_food()
        self.speed = self.SPEEDS["慢"]
        self.running = True
        self.score = 0
        self.food_count = 0
        self.star_lifetime = 0
        self.star_food_available = True # 新增标志位, 表示是否可以生成星形食物

        pygame.init()
        self.screen = pygame.display.set_mode((self.WIDTH, self.HEIGHT))
        pygame.display.set_caption("贪吃蛇")
        self.clock = pygame.time.Clock()

    def generate_food(self):
        if self.star_food_available and self.continuous_regular_food >= random.ra
ndint(3, 5):
            food_type = 'star'
            food_pos = self.get_random_position()
            head_x, head_y = self.snake[0]
            food_x, food_y = food_pos
            distance = abs(food_x - head_x) // self.CELL_SIZE + abs(food_y - head
_y) // self.CELL_SIZE
            self.star_lifetime = max(distance * 8, 40)
            self.star_food_available = False # 星形食物生成后, 设置为不可用, 直
到下次重新生成
        else:
            food_type = 'circle'
            food_pos = self.get_random_position()

```

```
        return food_pos, food_type

    def get_random_position(self):
        return (random.randint(0, (self.WIDTH // self.CELL_SIZE) - 1) * self.CELL_SIZE,
                random.randint(0, (self.HEIGHT // self.CELL_SIZE) - 1) * self.CELL_SIZE)

    def handle_events(self):
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                self.running = False
            elif event.type == pygame.KEYDOWN:
                self.change_direction(event)
                self.change_speed(event)

    def change_direction(self, event):
        if event.key == pygame.K_UP and self.direction != Direction.DOWN:
            self.direction = Direction.UP
        elif event.key == pygame.K_DOWN and self.direction != Direction.UP:
            self.direction = Direction.DOWN
        elif event.key == pygame.K_LEFT and self.direction != Direction.RIGHT:
            self.direction = Direction.LEFT
        elif event.key == pygame.K_RIGHT and self.direction != Direction.LEFT:
            self.direction = Direction.RIGHT

    def change_speed(self, event):
        if event.key == pygame.K_1:
            self.speed = self.SPEEDS["慢"]
        elif event.key == pygame.K_2:
            self.speed = self.SPEEDS["中"]
        elif event.key == pygame.K_3:
            self.speed = self.SPEEDS["快"]

    def update(self):
        head_x, head_y = self.snake[0]
        direction_x, direction_y = self.direction.value
        new_head = (head_x + direction_x * self.CELL_SIZE, head_y + direction_y * self.CELL_SIZE)
        new_head = self.wrap_around(new_head)
        if new_head in self.snake:
            self.running = False
            return
        self.snake.insert(0, new_head)
        food_position, food_type = self.food
        if new_head == food_position:
```

```

        self.food_count += 1
        if food_type == 'circle':
            self.score += 1
            self.continuous_regular_food += 1
        elif food_type == 'star':
            self.score += 5
            self.continuous_regular_food = 0
            self.star_lifetime = 0
            self.star_food_available = True # 吃到星形食物后, 重新允许生成星
形食物
            self.food = self.generate_food()
        else:
            self.snake.pop()

        # 处理星形食物的生命周期
        if self.food[1] == 'star' and self.star_lifetime > 0:
            self.star_lifetime -= 1
            if self.star_lifetime == 0:
                self.food = self.generate_food()

    def wrap_around(self, position):
        x, y = position
        x %= self.WIDTH
        y %= self.HEIGHT
        return (x, y)

    def draw_star(self, center, size):
        star_points = [
            (center[0], center[1] - size),
            (center[0] + size * 0.2, center[1] - size * 0.3),
            (center[0] + size, center[1] - size * 0.3),
            (center[0] + size * 0.3, center[1]),
            (center[0] + size * 0.5, center[1] + size),
            (center[0], center[1] + size * 0.4),
            (center[0] - size * 0.5, center[1] + size),
            (center[0] - size * 0.3, center[1]),
            (center[0] - size, center[1] - size * 0.3),
            (center[0] - size * 0.2, center[1] - size * 0.3)
        ]
        pygame.draw.polygon(self.screen, random.choice(self.COLORS), star_points)

    def draw(self):
        self.screen.fill(self.WHITE)
        for segment in self.snake:
            pygame.draw.rect(self.screen, self.GREEN, (segment[0], segment[1], se
lf.CELL_SIZE, self.CELL_SIZE))

```

```

        food_position, food_type = self.food
        if food_type == 'circle':
            pygame.draw.circle(self.screen, self.RED, (food_position[0] + 10, food_position[1] + 10), 10)
        else:
            self.draw_star(food_position, self.CELL_SIZE)
            font = pygame.font.Font(pygame.font.match_font('simhei'), 24)
            text = font.render(f"得分: {self.score} | 食物: {self.food_count} | 五角星剩余时间: {self.star_lifetime}", True, self.BLACK)
            self.screen.blit(text, (self.WIDTH - 300, 10))
        pygame.display.flip()

    def run(self):
        while self.running:
            self.handle_events()
            self.update()
            self.draw()
            self.clock.tick(self.speed)
        pygame.quit()

if __name__ == "__main__":
    game = SnakeGame()
    game.run()

```

主要修改点：

1. `self.star_food_available`：这是一个新添加的标志位，表示当前是否可以生成星形食物。每次星形食物被吃掉后，将其设置为 `False`，直到下一次生成食物时才有可能重新生成星形食物。
2. 生成食物时的逻辑：在 `generate_food()` 中，如果星形食物可用且满足一定条件时，才生成星形食物；否则生成普通圆形食物。

现在，只有在吃到星形食物后才会允许再次生成星形食物，而星形食物不会立即再次出现。

v1.12

显示速度的功能又丢了。让它添加吧。

v1.13

目前手工处理了一下，代码如下。

```
import pygame
import random
from enum import Enum

class Direction(Enum):
    UP = (0, -1)
    DOWN = (0, 1)
    LEFT = (-1, 0)
    RIGHT = (1, 0)

class SnakeGame:
    WIDTH, HEIGHT = 600, 400
    CELL_SIZE = 20
    WHITE = (255, 255, 255)
    GREEN = (0, 255, 0)
    RED = (255, 0, 0)
    BLACK = (0, 0, 0)
    COLORS = [(255, 215, 0), (255, 0, 255), (0, 255, 255), (255, 69, 0), (0, 255, 0)]

    SPEEDS = {"慢": 5, "中": 10, "快": 15}

    def __init__(self):
        self.continuous_regular_food = 0
        self.star_food_available = True # 新增标志位, 表示是否可以生成星形食物
        self.snake = [(100, 100), (90, 100), (80, 100)]
        self.direction = Direction.RIGHT
        self.food = self.generate_food()
        self.speed = self.SPEEDS["慢"]
        self.running = True
        self.score = 0
        self.food_count = 0
        self.star_lifetime = 0

        pygame.init()
        self.screen = pygame.display.set_mode((self.WIDTH, self.HEIGHT))
        pygame.display.set_caption("贪吃蛇")
        self.clock = pygame.time.Clock()

    def generate_food(self):
        if self.star_food_available and self.continuous_regular_food >= random.randint(3, 5):
            food_type = 'star'
```

```

        food_pos = self.get_random_position()
        head_x, head_y = self.snake[0]
        food_x, food_y = food_pos
        distance = abs(food_x - head_x) // self.CELL_SIZE + abs(food_y - head
_y) // self.CELL_SIZE
        self.star_lifetime = max(distance * 8, 40)
        self.star_food_available = False # 星形食物生成后, 设置为不可用, 直
到下次重新生成
    else:
        food_type = 'circle'
        food_pos = self.get_random_position()
    return food_pos, food_type

    def get_random_position(self):
        return (random.randint(0, (self.WIDTH // self.CELL_SIZE) - 1) * self.CELL
_SIZE,
                random.randint(0, (self.HEIGHT // self.CELL_SIZE) - 1) * self.CEL
L_SIZE)

    def handle_events(self):
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                self.running = False
            elif event.type == pygame.KEYDOWN:
                self.change_direction(event)
                self.change_speed(event)

    def change_direction(self, event):
        if event.key == pygame.K_UP and self.direction != Direction.DOWN:
            self.direction = Direction.UP
        elif event.key == pygame.K_DOWN and self.direction != Direction.UP:
            self.direction = Direction.DOWN
        elif event.key == pygame.K_LEFT and self.direction != Direction.RIGHT:
            self.direction = Direction.LEFT
        elif event.key == pygame.K_RIGHT and self.direction != Direction.LEFT:
            self.direction = Direction.RIGHT

    def change_speed(self, event):
        if event.key == pygame.K_1:
            self.speed = self.SPEEDS["慢"]
        elif event.key == pygame.K_2:
            self.speed = self.SPEEDS["中"]
        elif event.key == pygame.K_3:
            self.speed = self.SPEEDS["快"]

    def update(self):

```

```

        head_x, head_y = self.snake[0]
        direction_x, direction_y = self.direction.value
        new_head = (head_x + direction_x * self.CELL_SIZE, head_y + direction_y *
self.CELL_SIZE)
        new_head = self.wrap_around(new_head)
        if new_head in self.snake:
            self.running = False
            return
        self.snake.insert(0, new_head)
        food_position, food_type = self.food
        if new_head == food_position:
            self.food_count += 1
            if food_type == 'circle':
                self.score += 1
                self.continuous_regular_food += 1
            elif food_type == 'star':
                self.score += 5
                self.continuous_regular_food = 0
                self.star_lifetime = 0
                self.star_food_available = True # 吃到星形食物后, 重新允许生成星
形食物
        self.food = self.generate_food()
    else:
        self.snake.pop()

    # 处理星形食物的生命周期
    if self.food[1] == 'star' and self.star_lifetime > 0:
        self.star_lifetime -= 1
        if self.star_lifetime == 0:
            self.food = self.generate_food()

    def wrap_around(self, position):
        x, y = position
        x %= self.WIDTH
        y %= self.HEIGHT
        return (x, y)

    def draw_star(self, center, size):
        star_points = [
            (center[0], center[1] - size),
            (center[0] + size * 0.2, center[1] - size * 0.3),
            (center[0] + size, center[1] - size * 0.3),
            (center[0] + size * 0.3, center[1]),
            (center[0] + size * 0.5, center[1] + size),
            (center[0], center[1] + size * 0.4),
            (center[0] - size * 0.5, center[1] + size),

```

```

        (center[0] - size * 0.3, center[1]),
        (center[0] - size, center[1] - size * 0.3),
        (center[0] - size * 0.2, center[1] - size * 0.3)
    ]
    pygame.draw.polygon(self.screen, random.choice(self.COLORS), star_points)

    def draw(self):
        self.screen.fill(self.WHITE)
        for segment in self.snake:
            pygame.draw.rect(self.screen, self.GREEN, (segment[0], segment[1], self.CELL_SIZE, self.CELL_SIZE))
        food_position, food_type = self.food
        if food_type == 'circle':
            pygame.draw.circle(self.screen, self.RED, (food_position[0] + 10, food_position[1] + 10), 10)
        else:
            self.draw_star(food_position, self.CELL_SIZE)
        font = pygame.font.Font(pygame.font.match_font('simhei'), 20)
        text = font.render(f"得分:{self.score} | 食物:{self.food_count}", True, self.BLACK)
        self.screen.blit(text, (self.WIDTH - 300, 10))

        if not self.star_food_available and self.star_lifetime > 0:
            star_lifetime_text = font.render(f"五角星剩余时间: {self.star_lifetime}", True, self.BLACK)
            self.screen.blit(star_lifetime_text, (self.WIDTH - 300, 40))

        # 新增一行显示当前速度
        speed_text = font.render(f"速度: {'慢' if self.speed == self.SPEEDS['慢'] else '中' if self.speed == self.SPEEDS['中'] else '快'} (按1,2,3切换)", True, self.BLACK)
        self.screen.blit(speed_text, (10, 10))

    pygame.display.flip()

    def run(self):
        while self.running:
            self.handle_events()
            self.update()
            self.draw()
            self.clock.tick(self.speed)
        pygame.quit()

if __name__ == "__main__":

```

```
game = SnakeGame()
game.run()
```

v1.14

我：我给你一段代码，你先理解，回复明白，然后我提出新的需求。

它：明白

我：在现在代码的基础上，添加显示fps功能

它：

要在当前代码中添加显示 FPS（每秒帧数）功能，我们可以通过 `pygame.time.Clock()` 提供的 `get_fps()` 方法来获取 FPS 并在屏幕上显示。

以下是修改的部分：

1. 获取 FPS：在 `update()` 方法中或 `draw()` 方法中获取当前的 FPS。
2. 显示 FPS：在 `draw()` 方法中添加一行代码，用来显示 FPS。

修改后的代码如下：

```
import pygame
import random
from enum import Enum

class Direction(Enum):
    UP = (0, -1)
    DOWN = (0, 1)
    LEFT = (-1, 0)
    RIGHT = (1, 0)

class SnakeGame:
    WIDTH, HEIGHT = 600, 400
    CELL_SIZE = 20
    WHITE = (255, 255, 255)
    GREEN = (0, 255, 0)
    RED = (255, 0, 0)
    BLACK = (0, 0, 0)
    COLORS = [(255, 215, 0), (255, 0, 255), (0, 255, 255), (255, 69, 0), (0, 255, 0)]

    SPEEDS = {"慢": 5, "中": 10, "快": 15}

    def __init__(self):
        self.continuous_regular_food = 0
```

```
self.star_food_available = True # 新增标志位, 表示是否可以生成星形食物
self.snake = [(100, 100), (90, 100), (80, 100)]
self.direction = Direction.RIGHT
self.food = self.generate_food()
self.speed = self.SPEEDS["慢"]
self.running = True
self.score = 0
self.food_count = 0
self.star_lifetime = 0

pygame.init()
self.screen = pygame.display.set_mode((self.WIDTH, self.HEIGHT))
pygame.display.set_caption("贪吃蛇")
self.clock = pygame.time.Clock()

def generate_food(self):
    if self.star_food_available and self.continuous_regular_food >= random.ra
ndint(3, 5):
        food_type = 'star'
        food_pos = self.get_random_position()
        head_x, head_y = self.snake[0]
        food_x, food_y = food_pos
        distance = abs(food_x - head_x) // self.CELL_SIZE + abs(food_y - head
_y) // self.CELL_SIZE
        self.star_lifetime = max(distance * 8, 40)
        self.star_food_available = False # 星形食物生成后, 设置为不可用, 直
到下次重新生成
    else:
        food_type = 'circle'
        food_pos = self.get_random_position()
    return food_pos, food_type

def get_random_position(self):
    return (random.randint(0, (self.WIDTH // self.CELL_SIZE) - 1) * self.CELL
_SIZE,
            random.randint(0, (self.HEIGHT // self.CELL_SIZE) - 1) * self.CEL
L_SIZE)

def handle_events(self):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            self.running = False
        elif event.type == pygame.KEYDOWN:
            self.change_direction(event)
            self.change_speed(event)
```

```

def change_direction(self, event):
    if event.key == pygame.K_UP and self.direction != Direction.DOWN:
        self.direction = Direction.UP
    elif event.key == pygame.K_DOWN and self.direction != Direction.UP:
        self.direction = Direction.DOWN
    elif event.key == pygame.K_LEFT and self.direction != Direction.RIGHT:
        self.direction = Direction.LEFT
    elif event.key == pygame.K_RIGHT and self.direction != Direction.LEFT:
        self.direction = Direction.RIGHT

def change_speed(self, event):
    if event.key == pygame.K_1:
        self.speed = self.SPEEDS["慢"]
    elif event.key == pygame.K_2:
        self.speed = self.SPEEDS["中"]
    elif event.key == pygame.K_3:
        self.speed = self.SPEEDS["快"]

def update(self):
    head_x, head_y = self.snake[0]
    direction_x, direction_y = self.direction.value
    new_head = (head_x + direction_x * self.CELL_SIZE, head_y + direction_y *
self.CELL_SIZE)
    new_head = self.wrap_around(new_head)
    if new_head in self.snake:
        self.running = False
        return
    self.snake.insert(0, new_head)
    food_position, food_type = self.food
    if new_head == food_position:
        self.food_count += 1
        if food_type == 'circle':
            self.score += 1
            self.continuous_regular_food += 1
        elif food_type == 'star':
            self.score += 5
            self.continuous_regular_food = 0
            self.star_lifetime = 0
            self.star_food_available = True # 吃到星形食物后, 重新允许生成星
形食物
        self.food = self.generate_food()
    else:
        self.snake.pop()

# 处理星形食物的生命周期
if self.food[1] == 'star' and self.star_lifetime > 0:

```

```

        self.star_lifetime -= 1
        if self.star_lifetime == 0:
            self.food = self.generate_food()

    def wrap_around(self, position):
        x, y = position
        x %= self.WIDTH
        y %= self.HEIGHT
        return (x, y)

    def draw_star(self, center, size):
        star_points = [
            (center[0], center[1] - size),
            (center[0] + size * 0.2, center[1] - size * 0.3),
            (center[0] + size, center[1] - size * 0.3),
            (center[0] + size * 0.3, center[1]),
            (center[0] + size * 0.5, center[1] + size),
            (center[0], center[1] + size * 0.4),
            (center[0] - size * 0.5, center[1] + size),
            (center[0] - size * 0.3, center[1]),
            (center[0] - size, center[1] - size * 0.3),
            (center[0] - size * 0.2, center[1] - size * 0.3)
        ]
        pygame.draw.polygon(self.screen, random.choice(self.COLORS), star_points)

    def draw(self):
        self.screen.fill(self.WHITE)
        for segment in self.snake:
            pygame.draw.rect(self.screen, self.GREEN, (segment[0], segment[1], self.CELL_SIZE, self.CELL_SIZE))
        food_position, food_type = self.food
        if food_type == 'circle':
            pygame.draw.circle(self.screen, self.RED, (food_position[0] + 10, food_position[1] + 10), 10)
        else:
            self.draw_star(food_position, self.CELL_SIZE)
        font = pygame.font.Font(pygame.font.match_font('simhei'), 20)
        text = font.render(f"得分:{self.score} | 食物:{self.food_count}", True, self.BLACK)
        self.screen.blit(text, (self.WIDTH - 300, 10))

        if not self.star_food_available and self.star_lifetime > 0:
            star_lifetime_text = font.render(f"五角星剩余时间: {self.star_lifetime}", True, self.BLACK)
            self.screen.blit(star_lifetime_text, (self.WIDTH - 300, 40))

```

```

        # 显示当前速度
        speed_text = font.render(f"速度: {'慢' if self.speed == self.SPEEDS['慢']
else '中' if self.speed == self.SPEEDS['中'] else '快'} (按1,2,3切换)", True, sel
f.BLACK)
        self.screen.blit(speed_text, (10, 10))

        # 显示FPS
        fps = int(self.clock.get_fps())
        fps_text = font.render(f"FPS: {fps}", True, self.BLACK)
        self.screen.blit(fps_text, (10, 40))

    pygame.display.flip()

    def run(self):
        while self.running:
            self.handle_events()
            self.update()
            self.draw()
            self.clock.tick(self.speed)
        pygame.quit()

if __name__ == "__main__":
    game = SnakeGame()
    game.run()

```

关键更改：

1. 获取 FPS：在 `draw()` 方法中，使用 `fps = int(self.clock.get_fps())` 获取当前的 FPS。
2. 显示 FPS：通过 `font.render(f"FPS: {fps}", True, self.BLACK)` 在屏幕上显示 FPS。

现在，游戏界面将会显示当前的 FPS，并根据游戏运行的流畅度动态更新。

```
self.clock.tick(30) # 固定fps为30
```

但是fps调大后出现蛇移动太快的的问题，添加移动因子又太慢了。需要多次问gpt。

目前代码是

```

import pygame
import random
from enum import Enum

```

```

class Direction(Enum):
    UP = (0, -1)
    DOWN = (0, 1)
    LEFT = (-1, 0)
    RIGHT = (1, 0)

class SnakeGame:
    WIDTH, HEIGHT = 600, 400
    CELL_SIZE = 20
    WHITE = (255, 255, 255)
    GREEN = (0, 255, 0)
    RED = (255, 0, 0)
    BLACK = (0, 0, 0)
    COLORS = [(255, 215, 0), (255, 0, 255), (0, 255, 255), (255, 69, 0), (0, 255,
0)]

    SPEEDS = {"慢": 5, "中": 10, "快": 15}

    def __init__(self):
        self.continuous_regular_food = 0
        self.star_food_available = True # 新增标志位, 表示是否可以生成星形食物
        self.snake = [(100, 100), (90, 100), (80, 100)]
        self.direction = Direction.RIGHT
        self.food = self.generate_food()
        self.speed = self.SPEEDS["慢"]
        self.running = True
        self.score = 0
        self.food_count = 0
        self.star_lifetime = 0
        self.fps = 30 # 固定FPS为30
        self.move_speed_factor = self.speed / 9.0 # 用一个比例因子调整蛇的移动速
度

        pygame.init()
        self.screen = pygame.display.set_mode((self.WIDTH, self.HEIGHT))
        pygame.display.set_caption("贪吃蛇")
        self.clock = pygame.time.Clock()

    def generate_food(self):
        if self.star_food_available and self.continuous_regular_food >= random.ra
ndint(3, 5):
            food_type = 'star'
            food_pos = self.get_random_position()
            head_x, head_y = self.snake[0]
            food_x, food_y = food_pos

```

```
        distance = abs(food_x - head_x) // self.CELL_SIZE + abs(food_y - head
_y) // self.CELL_SIZE
        self.star_lifetime = max(distance * 8, 40)
        self.star_food_available = False # 星形食物生成后, 设置为不可用, 直
到下次重新生成
    else:
        food_type = 'circle'
        food_pos = self.get_random_position()
        return food_pos, food_type

    def get_random_position(self):
        return (random.randint(0, (self.WIDTH // self.CELL_SIZE) - 1) * self.CELL
_SIZE,
                random.randint(0, (self.HEIGHT // self.CELL_SIZE) - 1) * self.CEL
L_SIZE)

    def handle_events(self):
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                self.running = False
            elif event.type == pygame.KEYDOWN:
                self.change_direction(event)
                self.change_speed(event)

    def change_direction(self, event):
        if event.key == pygame.K_UP and self.direction != Direction.DOWN:
            self.direction = Direction.UP
        elif event.key == pygame.K_DOWN and self.direction != Direction.UP:
            self.direction = Direction.DOWN
        elif event.key == pygame.K_LEFT and self.direction != Direction.RIGHT:
            self.direction = Direction.LEFT
        elif event.key == pygame.K_RIGHT and self.direction != Direction.LEFT:
            self.direction = Direction.RIGHT

    def change_speed(self, event):
        if event.key == pygame.K_1:
            self.speed = self.SPEEDS["慢"]
        elif event.key == pygame.K_2:
            self.speed = self.SPEEDS["中"]
        elif event.key == pygame.K_3:
            self.speed = self.SPEEDS["快"]

    def update(self):
        head_x, head_y = self.snake[0]
        direction_x, direction_y = self.direction.value
        new_head = (head_x + direction_x * self.CELL_SIZE * self.move_speed_facto
```

```

r, head_y + direction_y * self.CELL_SIZE * self.move_speed_factor)
    new_head = self.wrap_around(new_head)
    if new_head in self.snake:
        self.running = False
        return
    self.snake.insert(0, new_head)
    food_position, food_type = self.food
    if new_head == food_position:
        self.food_count += 1
        if food_type == 'circle':
            self.score += 1
            self.continuous_regular_food += 1
        elif food_type == 'star':
            self.score += 5
            self.continuous_regular_food = 0
            self.star_lifetime = 0
            self.star_food_available = True # 吃到星形食物后, 重新允许生成星
形食物
        self.food = self.generate_food()
    else:
        self.snake.pop()

    # 处理星形食物的生命周期
    if self.food[1] == 'star' and self.star_lifetime > 0:
        self.star_lifetime -= 1
        if self.star_lifetime == 0:
            self.food = self.generate_food()

def wrap_around(self, position):
    x, y = position
    x %= self.WIDTH
    y %= self.HEIGHT
    return (x, y)

def draw_star(self, center, size):
    star_points = [
        (center[0], center[1] - size),
        (center[0] + size * 0.2, center[1] - size * 0.3),
        (center[0] + size, center[1] - size * 0.3),
        (center[0] + size * 0.3, center[1]),
        (center[0] + size * 0.5, center[1] + size),
        (center[0], center[1] + size * 0.4),
        (center[0] - size * 0.5, center[1] + size),
        (center[0] - size * 0.3, center[1]),
        (center[0] - size, center[1] - size * 0.3),
        (center[0] - size * 0.2, center[1] - size * 0.3)

```

```

    ]
    pygame.draw.polygon(self.screen, random.choice(self.COLORS), star_points)

    def draw(self):
        self.screen.fill(self.WHITE)
        for segment in self.snake:
            pygame.draw.rect(self.screen, self.GREEN, (segment[0], segment[1], self.CELL_SIZE, self.CELL_SIZE))
        food_position, food_type = self.food
        if food_type == 'circle':
            pygame.draw.circle(self.screen, self.RED, (food_position[0] + 10, food_position[1] + 10), 10)
        else:
            self.draw_star(food_position, self.CELL_SIZE)
        font = pygame.font.Font(pygame.font.match_font('simhei'), 20)
        text = font.render(f"得分:{self.score} | 食物:{self.food_count}", True, self.BLACK)
        self.screen.blit(text, (self.WIDTH - 300, 10))

        if not self.star_food_available and self.star_lifetime > 0:
            star_lifetime_text = font.render(f"五角星剩余时间: {self.star_lifetime}", True, self.BLACK)
            self.screen.blit(star_lifetime_text, (self.WIDTH - 300, 40))

        # 显示当前速度
        speed_text = font.render(f"速度: {'慢' if self.speed == self.SPEEDS['慢'] else '中' if self.speed == self.SPEEDS['中'] else '快'} (按1,2,3切换)", True, self.BLACK)
        self.screen.blit(speed_text, (10, 10))

        # 显示FPS
        fps = int(self.clock.get_fps())
        fps_text = font.render(f"FPS: {fps}", True, self.BLACK)
        self.screen.blit(fps_text, (10, 40))

    pygame.display.flip()

    def run(self):
        while self.running:
            self.handle_events()
            self.update()
            self.draw()
            # self.clock.tick(self.speed)
            self.clock.tick(self.fps) # 设置为固定的FPS为30

        pygame.quit()

```

```
if __name__ == "__main__":
    game = SnakeGame()
    game.run()
```

2.0

将v1.6中的贪吃蛇游戏改造成libgdx实现一下

我：将上面的贪吃蛇游戏改造成libgdx实现一下

它：将上述贪吃蛇游戏从 `pygame` 改造成使用 `LibGDX` 的实现，涉及到的关键步骤包括：

1. 设置 LibGDX 项目：首先，您需要设置一个 LibGDX 项目。可以使用官方的 [LibGDX Setup Tool](#) 来创建一个新的项目。
2. 替换 Pygame 相关代码：我们需要将 `pygame` 中的渲染、输入、事件处理等功能转换为 LibGDX 中的实现方法。
3. 游戏结构转换：LibGDX 有自己的游戏结构和生命周期管理，所有的游戏逻辑将被移动到 `ApplicationListener`（通常是 `Game` 或 `Screen` 类中）。

代码实现

以下是将您的贪吃蛇游戏转换为 LibGDX 实现的代码：

```
import com.badlogic.gdx.ApplicationAdapter;
import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.graphics.Color;
import com.badlogic.gdx.graphics.GL20;
import com.badlogic.gdx.graphics.Texture;
import com.badlogic.gdx.graphics.g2d.SpriteBatch;
import com.badlogic.gdx.Input;
import com.badlogic.gdx.math.Rectangle;
import java.util.LinkedList;
import java.util.Random;

public class SnakeGame extends ApplicationAdapter {
    private SpriteBatch batch;
    private LinkedList<Rectangle> snake;
    private Rectangle food;
    private int cellSize = 20;
    private int speed = 5;
    private int direction = Input.Keys.RIGHT;
    private boolean gameOver = false;
```

```

private String speedText;

@Override
public void create() {
    batch = new SpriteBatch();
    snake = new LinkedList<>();
    snake.add(new Rectangle(100, 100, cellSize, cellSize));
    snake.add(new Rectangle(80, 100, cellSize, cellSize));
    snake.add(new Rectangle(60, 100, cellSize, cellSize));
    spawnFood();
    speedText = "Speed: Slow (Press 1, 2, 3 to change)";
}

@Override
public void render() {
    if (gameOver) {
        return;
    }

    // Clear screen
    Gdx.gl.glClearColor(GL20.GL_COLOR_BUFFER_BIT);

    // Handle user input
    handleInput();

    // Update snake position
    update();

    // Draw everything
    batch.begin();
    batch.setColor(Color.WHITE);

    // Draw snake
    for (Rectangle segment : snake) {
        batch.draw(new Texture("square.png"), segment.x, segment.y, cellSize,
cellSize); // Draw each segment
    }

    // Draw food (red circle)
    batch.setColor(Color.RED);
    batch.draw(new Texture("circle.png"), food.x, food.y, cellSize, cellSize);

    // Display speed
    // In LibGDX, you typically use BitmapFont to display text, but it's simplified here.

```

```
        batch.end();
    }

    private void handleInput() {
        if (Gdx.input.isKeyPressed(Input.Keys.UP) && direction != Input.Keys.DOWN) {
            direction = Input.Keys.UP;
        }
        if (Gdx.input.isKeyPressed(Input.Keys.DOWN) && direction != Input.Keys.UP) {
            direction = Input.Keys.DOWN;
        }
        if (Gdx.input.isKeyPressed(Input.Keys.LEFT) && direction != Input.Keys.RIGHT) {
            direction = Input.Keys.LEFT;
        }
        if (Gdx.input.isKeyPressed(Input.Keys.RIGHT) && direction != Input.Keys.LEFT) {
            direction = Input.Keys.RIGHT;
        }

        if (Gdx.input.isKeyPressed(Input.Keys.NUM_1)) {
            speed = 5;
            speedText = "Speed: Slow";
        } else if (Gdx.input.isKeyPressed(Input.Keys.NUM_2)) {
            speed = 10;
            speedText = "Speed: Medium";
        } else if (Gdx.input.isKeyPressed(Input.Keys.NUM_3)) {
            speed = 15;
            speedText = "Speed: Fast";
        }
    }

    private void update() {
        if (gameOver) {
            return;
        }

        // Move the snake
        Rectangle head = snake.getFirst();
        Rectangle newHead = new Rectangle(head.x, head.y, cellSize, cellSize);

        if (direction == Input.Keys.UP) newHead.y -= cellSize;
        if (direction == Input.Keys.DOWN) newHead.y += cellSize;
        if (direction == Input.Keys.LEFT) newHead.x += cellSize;
        if (direction == Input.Keys.RIGHT) newHead.x -= cellSize;
    }
}
```

```

        // Wrap around the screen
        if (newHead.x < 0) newHead.x = Gdx.graphics.getWidth() - cellSize;
        if (newHead.x >= Gdx.graphics.getWidth()) newHead.x = 0;
        if (newHead.y < 0) newHead.y = Gdx.graphics.getHeight() - cellSize;
        if (newHead.y >= Gdx.graphics.getHeight()) newHead.y = 0;

        // Check collision with self
        if (snake.contains(newHead)) {
            gameOver = true;
        }

        snake.addFirst(newHead);

        // Check if food is eaten
        if (newHead.overlaps(food)) {
            spawnFood();
        } else {
            snake.removeLast();
        }
    }

    private void spawnFood() {
        Random rand = new Random();
        int x = rand.nextInt(Gdx.graphics.getWidth() / cellSize) * cellSize;
        int y = rand.nextInt(Gdx.graphics.getHeight() / cellSize) * cellSize;
        food = new Rectangle(x, y, cellSize, cellSize);
    }

    @Override
    public void resize(int width, int height) {
        // Not required in this case, but can be used for screen resizing.
    }

    @Override
    public void hide() {
        // Cleanup when the game is paused or minimized.
    }

    @Override
    public void pause() {
        // Pause the game, could save state here.
    }

    @Override
    public void resume() {

```

```

        // Resuming the game if it was paused.
    }

    @Override
    public void dispose() {
        batch.dispose();
        // Clean up other resources like textures or sounds
    }
}

```

说明：

1. 渲染和输入：游戏的绘制和输入处理已经被移植到 `LibGDX` 的 `render()` 和 `handleInput()` 方法中。用户的键盘输入被通过 `Gdx.input.isKeyPressed()` 来检测。
2. 蛇的移动：蛇的位置通过 `LinkedList` 存储，每次蛇头移动时，新头部会被插入列表的开头，蛇尾被移除，形成蛇的前进效果。
3. 食物：食物被绘制为一个红色的矩形，您可以根据需求将其改成圆形。此处采用的是矩形绘制方法，实际使用时您可以通过 `batch.draw()` 绘制图形来模拟圆形。
4. 背景色：游戏窗口背景色设置为白色，`Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT)` 在每一帧的渲染开始时清除屏幕。
5. 游戏速度调整：通过按数字键 `1`，`2`，`3` 切换游戏的速度。
6. 碰撞检测和游戏结束：当蛇头碰到自己或墙壁时，游戏结束。

设置 LibGDX 项目：

1. 通过 [LibGDX Setup Tool](#) 创建一个新的项目。
2. 将 `SnakeGame.java` 类添加到项目中，并确保相关的资源（例如食物的纹理）可以正确加载。
3. 在 `core` 模块中设置主类为 `SnakeGame`。

总结：

上述代码实现了一个基本的贪吃蛇游戏，您可以通过键盘控制蛇的移动，改变游戏速度，且当蛇碰到墙壁时，它会从另一边出来。食物使用矩形（可以改为圆形）绘制。如果需要更多自定义或改进，您可以进一步扩展该实现。

下载[libGDX Project Setup Tool (gdx-liftoff)](<https://libgdx.com/wiki/start/project-generation>)，下载文件是 <https://github.com/libgdx/gdx-liftoff/releases/tag/v1.13.1.2> 的 `gdx-liftoff-winX64.zip`。解压并双击 `gdx-liftoff.exe`，填写相关信息就能生成工程了。

ollama

this post was written in

info

official website: [Ollama](#)

download and install

download exe file or use powershell command(I use exe file) and install.

on 2026-03-05.

chat

please change 'model location' before downloading model, do not use `c:/` .

choose a model (qwen3:4b or search other model qwen3:0.6b) and ask him.

api

search model: <https://ollama.com/library/qwen3>

chat(ChatGPT式对话)

```
curl -X POST 'http://localhost:11434/api/chat' -H 'Content-Type: application/json' -d '{ "model": "qwen3:4b", "think1": false, "stream": false, "messages": [{"role": "user", "content": "你是谁, 用一句话回答"}] }'
```

不要使用think参数，会导致命令行一直得不到返回。

generate(单 prompt 生成)

```
curl -X POST 'http://localhost:11434/api/generate' -H 'Content-Type: application/json' -d '{ "model": "qwen3:4b", "prompt": "写一句赞美春天的话", "stream1": false }'
```

不要使用stream1参数，会导致命令行一直得不到返回.

[out-of-date]Windows10本地使用ollama和chatbox运行deepseek

This post is out of date, written on 2025-03-19.

Windows10本地使用ollama和chatbox运行deepseek模型

步骤一：下载Ollama

打开 [Ollama](#)，下载windows版，等待下载完成。之后双击安装包，一直下一步就安装成功了。



Get up and running with large language models.

Run [Llama 3.3](#), [DeepSeek-R1](#), [Phi-4](#), [Mistral](#), [Gemma 3](#), and other models, locally.

[Download ↓](#)

Available for macOS,
Linux, and Windows

Download Ollama



macOS



Linux



Windows

[Download for Windows](#)

Requires Windows 10 or later

步骤二：下载deepseek-r1模型

访问[deepseek-r1模型页面](#)，选择自己需要的版本，此处使用最小的。

deepseek-r1

DeepSeek's first-generation of reasoning models with comparable performance to OpenAI-o1, including six dense models distilled from DeepSeek-R1 based on Llama and Qwen.

[1.5b](#) [7b](#) [8b](#) [14b](#) [32b](#) [70b](#) [671b](#)

↓ 28.1M Pulls ⌚ Updated 5 weeks ago

Model Size	Size
1.5b	1.1GB
7b	4.7GB
8b	4.9GB
14b	9.0GB
32b	20GB
70b	43GB
671b	404GB

Model ID	Size
a42b25d8c10a	1.1GB

ollama run deepseek-r1:1.5b

```
parameters 1.78B · quantization Q4_K_M
```

```
begin_of_sentence | >", "< | end_of_sentence | >",
```

```
}} {{ .System }} {{ end }} {{- range $i, $_ := .Mes...
```

```
opyright (c) 2023 DeepSeek Permission is hereby gra...
```

在powershell中执行 `ollama run deepseek-r1:1.5b`

需要花点时间，等看到success就成功了。

我们可以在命令行里问它一些问题。

```
PS C:\Users\Administrator> ollama list
NAME          ID          SIZE      MODIFIED
deepseek-r1:1.5b a42b25d8c10a 1.1 GB    10 hours ago
PS C:\Users\Administrator> ollama run deepseek-r1:1.5b
>>> 你是谁?
<think>
</think>
您好！我是由中国的深度求索（DeepSeek）公司开发的智能助手DeepSeek-R1。如您有任何任何问题，我会尽我所能为您提供帮助。
>>> Send a message (? for help)
```

步骤三：使用chatbox

打开[Chatbox AI官网](#)，点击免费下载。



Chatbox AI 是一款 AI 客户端应用和智能助手，支持众多先进的 AI 模型和 API，可在 Windows、MacOS、Android、iOS、Linux 和网页版上使用。

Chatbox 软件有多种用途，但作为一个模型 API 和本地模型的工具，其主要功能一直都是完全免费的。近期出现了附带 Chatbox 的所谓**一键本地部署 DeepSeek 的付费捆绑软件安装包**，这类软件均与 Chatbox 无关。在 Chatbox 用户条款中明确禁止这一捆绑销售的侵权行为。请用户谨慎辨别，谨防上当受骗！

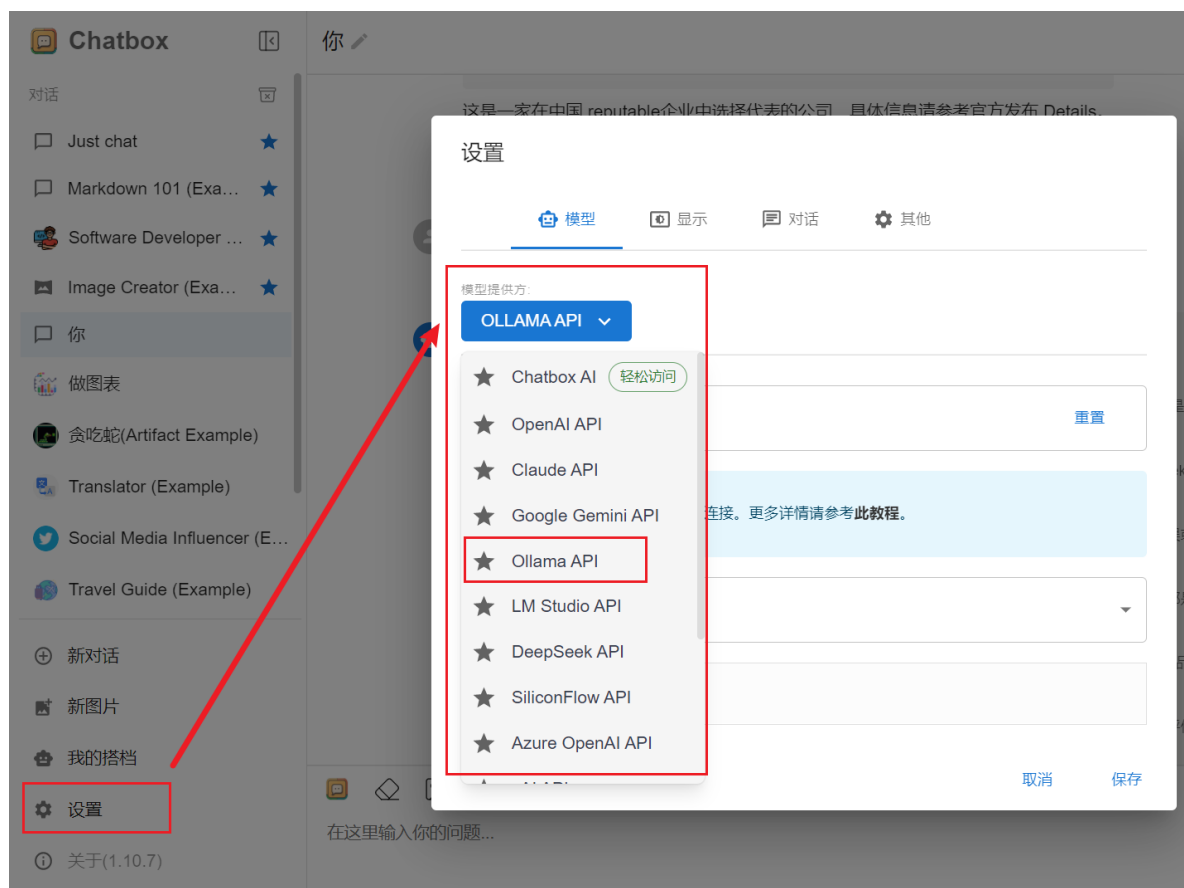
↓ 免费下载 (for Windows) 启动网页版

获取移动版

iOS Android APK + 更多选项

下载完成之后双击安装包，一直下一步就安装成功。

配置ollama



设置

模型提供方:

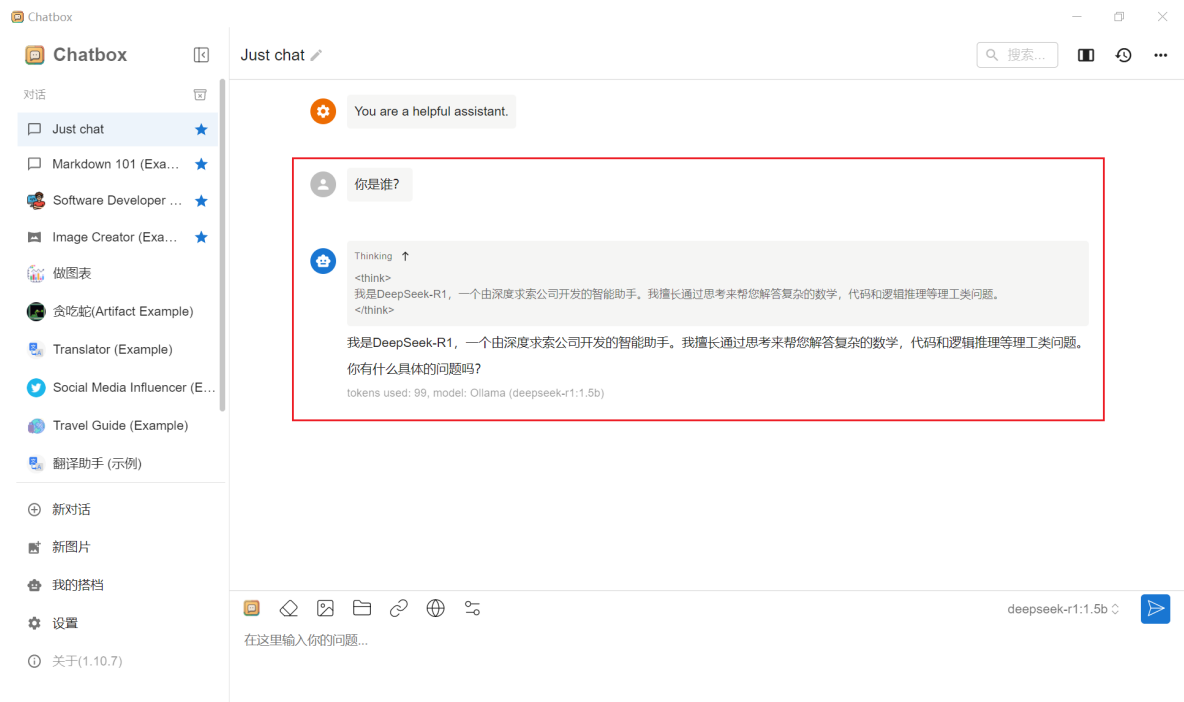
- OLLAMA API
- ★ Chatbox AI (轻松访问)
- ★ OpenAI API
- ★ Claude API
- ★ Google Gemini API
- ★ Ollama API
- ★ LM Studio API
- ★ DeepSeek API
- ★ SiliconFlow API
- ★ Azure OpenAI API

取消 保存

配置deepseek模型，此时我们只有一个。



再问它“你是谁？”



结束。

AI绘图

##

comfyUI

ComfyUI安装与使用

安装

```
git clone git@github.com:comfyanonymous/ComfyUI.git
```

```
#官方建议 安装python-v3.12, 我用的是v3.10
```

```
pip install -r requirements.txt
```

```
python main.py --disable-cuda-malloc --use-split-cross-attention --cpu
```

stable-diffusion-webui-getting-started

关键字（方便模糊搜索出本文）

- stable diffusion
- stable-diffusion
- stable-diffusion-webui
- stable diffusion webui
- 图片+语音转视频

说明

小新pro14 2023版本 32G
电脑 win10-lstc
版本 1809
操作系统内部版本：17763.4377

独显：intel iris Xe Graphics

开始

使用的git：<https://github.com/AUTOMATIC1111/stable-diffusion-webui.git>
commitId: baf6946e06249c5af9851c60171692c44ef633e0

git-scm：Git for Windows v2.37.3

安装python3.10.11，（不需要配置环境变量）假设安装目录是D:\ws\apps\Python310
修改 webui-user.bat，将PYTHON

```
set PYTHON="D:\ws\apps\Python310\Python.exe"
```

然后打开cmd，输入webui-user.bat，等待安装完成（可能需要翻q）。

访问 <http://127.0.0.1:7860>

安装SadTalker

附件

文件 : webui-user.bat

```
@echo off

set PYTHON="D:\ws\apps\Python310\Python.exe"
set GIT=
set VENV_DIR=
REM --xformers --skip-torch-cuda-test --precision full --no-half --disable-safe-unpickle
set COMMANDLINE_ARGS=--skip-torch-cuda-test --precision full --no-half --disable-safe-unpickle

call webui.bat
```

引用

- <http://127.0.0.1:7860/>
- <https://ivonblog.com/posts/windows-stable-diffusion-webui/>
- <https://www.bilibili.com/read/cv21987039>
- https://blog.csdn.net/weixin_43924605/article/details/130288081
- <https://blog.csdn.net/zfengfei/article/details/130265568>
- <https://zhuanlan.zhihu.com/p/613530403>
- <https://civitai.com/>
- <https://github.com/OpenTalker/SadTalker/issues/342>
 - Open the modules/ui_tempdir.py file and in line 34 replace “def save_pil_to_file(self, pil_image, dir=None):” with this: “def save_pil_to_file(self, pil_image, dir=None, format=“png”):”

stable-diffusion-webui

installation

stable-diffusion-webui

- Install Python [3.10.6](#) (Newer version of Python does not support torch), checking "Add Python to PATH".
- Install [git](#).
- Download the stable-diffusion-webui repository, for example by running `git clone https://github.com/AUTOMATIC1111/stable-diffusion-webui.git` or `git clone git@github.com:AUTOMATIC1111/stable-diffusion-webui.git`.
- Run `webui-user.bat` from Windows Explorer as normal, non-administrator, user. (It may cost much time(almost 20min~1hour))
 - if 'RuntimeError: Torch is not able to use GPU; add --skip-torch-cuda-test to COMMANDLINE_ARGS variable to disable this check' occurs, edit `webui-user.bat` and append `--skip-torch-cuda-test` to `COMMANDLINE_ARGS`. my runnable script is `set COMMANDLINE_ARGS= --precision full --no-half --skip-torch-cuda-test`. announce that there is a space after `=`.
 - there will be `http://127.0.0.1:7860` in console, which indicates a successful installation.

usage

visit `http://127.0.0.1:7860`.

- if you got `Expecting value: line 1 column 1 (char 0)` in home page or after click generate button. check and close your proxy.

model

目前个人理解，模型分为checkpoint和lora。前者比较大（1G以上），是底图，lora模型是用来微小微调的（比较小，100M以下）。

1、CKPT (CheckPoint)

经过训练的图片合集，被称作模型，也就是checkpoint，体积较大，一般真人版的单个模型的大小在7GB左右，动漫版的在2-5个G之间。早期的CKPT后缀名是ckpt，如今新的CKPT后缀名都是safetensors

2、Lora是一种体积较小的绘画模型，是对大模型的微调。与每次作画只能选择一个大模型不同，lora模型可以在已选择大模型的基础上添加一个甚至多个。一般体积在几十到几百兆左右。后缀也是safetensors.

checkpoint模型放到 stable-diffusion-webui 项目的 models/Stable-diffusion/目录下。

lora模型放到 stable-diffusion-webui 项目的 models/Lora/ 目录下。

<https://civitai.com/models>

checkpoint模型：

- [v1-5-pruned-emaonly sd-ui自带](#)
- [ChilloutMix](#) (需要登录)
- [majicMIX realistic 麦橘写实](#)

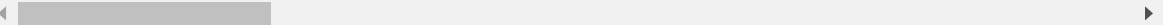
已尝试的模型：

- <https://civitai.com/models/15271/20d>
- <https://civitai.com/models/14171/cutegirlmix4>

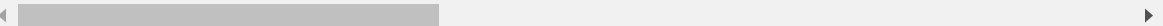
要放到 stable-diffusion-webui 项目的 models/Lora/ 目录下。之后就可以在 <http://127.0.0.1:7860/>的lora标签页下看到mix4

```
<lora:mix4:0.7>, mix4, <lora:20d:0.5>, 20d, solo, long_hair, shirt, dress, high_heels,full_body, look_at_view  
Negative prompt: paintings, sketches, (worst quality:2), (low quality:2), (normal quality:2), lowres, normal q
```

```
Size: 512x1024, Seed: 2144095954, Model: chilloutmix_NiPrunedFp32Fix, Steps: 30, Sampler: DPM++ SDE Karras, CF
```



```
masterpiece, best quality,realistic,1girl,(hanfu, pink short shan, gradient green pleated skirt, pink songmo)  
Negative prompt: nsfw,ng_deepnegative_v1_75t, badhandv4, (worst quality:2), (low quality:2), (normal quality:2)  
Steps: 25, CFG scale: 8, Sampler: DPM++ 2S a Karras, Seed: 506783545, Size: 640x768, Model: beautifulRealistic
```



https://www.4b3.com/item?a_model=stable-diffusion

liblib中需要付费才能下载模型（但是模型里的示例图片的提示词可以在sd中使用）：

- [Flux_小红书真实风格 | 日常照片 | 极致逼真-LoRA-AI老X-LiblibAI](#)

- [FLUX_小红书极致大胸美女【图文流量密码】-LoRA-别让唐僧跑了-LiblibAI](#)
- [LiblibAI-哩布哩布AI - 中国领先的AI创作平台](#)

goal

根据图片生成写真

AI摄影写真

油画

科幻画

复古

国风

<https://www.bilibili.com/video/BV16v2hYVEA5/>

常用插件

- [git@github.com:picobyte/stable-diffusion-webui-wd14-tagger.git](#) 反推图片中的标签
- <https://github.com/Physton/sd-webui-prompt-all-in-one>
 - [git@github.com:Physton/sd-webui-prompt-all-in-one.git](#)
- <https://github.com/TMElyralab/MuseTalk>
 - 数字人？
- [git@github.com:kabachuha/sd-webui-text2video.git](#)

Sd-webui-segment-anything | 自由换装

[git@github.com:facebookresearch/segment-anything.git](#)

api

<https://juejin.cn/post/7272730422735061044>

sd-webui-and-SadTalker

版本

- sd-webui(<https://github.com/AUTOMATIC1111/stable-diffusion-webui> version: v1.10.1)
- SadTalker(<https://github.com/OpenTalker/SadTalker/> commit: cd4c0465ae0b54a6f85af57f5c65fec9fe23e7f8)
- <https://github.com/OpenTalker/SadTalker.git>
 - `git@github.com:OpenTalker/SadTalker.git`
 - `set COMMANDLINE_ARGS="--disable-safe-unpickle"`
`set SADTALKER_CHECKPOINTS=D:\openai.wiki\stable-diffusion-webui\extensions\SadTalker\checkpoints`

下载 & 修改 & 使用

使用 `git clone https://github.com/OpenTalker/SadTalker` 下载，sd-webui在线安装会出问题。

下载好以后，复制到D:\ai\stable-diffusion-webui\extensions中，注意文件夹的名称必须是 `SadTalker`。

修改

修改文件名

requirements3d.txt

req.txt

requirements.txt

这三个文件，名称后加个1，别让自动安装依赖。

修改代码

```
return librosa.stft(y=y, n_fft=hp.n_fft, hop_length=get_hop_size(), win_length=hp.win_size)
在src/utils/audio.py搜索上面这行代码，改成下面的
return librosa.stft(y=y, n_fft=hp.n_fft, hop_length=get_hop_size(), win_length=hp.win_size, dtype=floa
```

```

在src\face3d\util\preprocess.py中
添加`from numpy import array`
搜索这行代码
    trans_params = np.array([w0, h0, s, t[0], t[1]])
改成它
    trans_params = array([float(w0), float(h0), float(s), float(t[0]), float(t[1])])

```

```

在src\face3d\util\my_awing_arch.py中
搜索这行代码
    preds = preds.astype(np.float, copy=False)
改成它
    preds = preds.astype(float, copy=False)

```

修改.py文件时注意空格对齐。

这样安装

```
venv\Scripts\pip.exe install librosa==0.9.2
```

报什么错安装什么，别一下安装很多。

下载模型文件：

放到D:\ai\stable-diffusion-webui\extensions\SadTalker\checkpoints中。

在webui-user.bat文件中的 `call webui.bat` 的前面添加 `set`
`SADTALKER_CHECKPOINTS=D:\ai\stable-diffusion-webui\extensions\SadTalker\checkpoints`

(更建议将这些模型文件放到stable-diffusion-webui或SadTalker目录之外，如 `D:\ai\sd-
 extensdion-files\SadTalker`，这样使用 `set SADTALKER_CHECKPOINTS=D:\ai\sd-
 extensdion-files\SadTalker\checkpoints` 即可，见
[官方文档](#))

<https://github.com/OpenTalker/SadTalker/releases>

在v0.0.2版本里

```

mapping_00109-model.pth.tar 149 MB ./checkpoints/mapping_00109-model.pth.tar
mapping_00229-model.pth.tar 148 MB ./checkpoints/mapping_00229-model.pth.tar
SadTalker_V0.0.2_256.safetensors 691 MB ./checkpoints/SadTalker_V0.0.2_256.safetensors
SadTalker_V0.0.2_512.safetensors 691 MB ./checkpoints/SadTalker_V0.0.2_512.safetensors

```

```

auido2exp_00300-model.pth
auido2pose_00140-model.pth
epoch_20.pth
wav2lip.pth
facevid2vid_00189-model.pth.tar

```

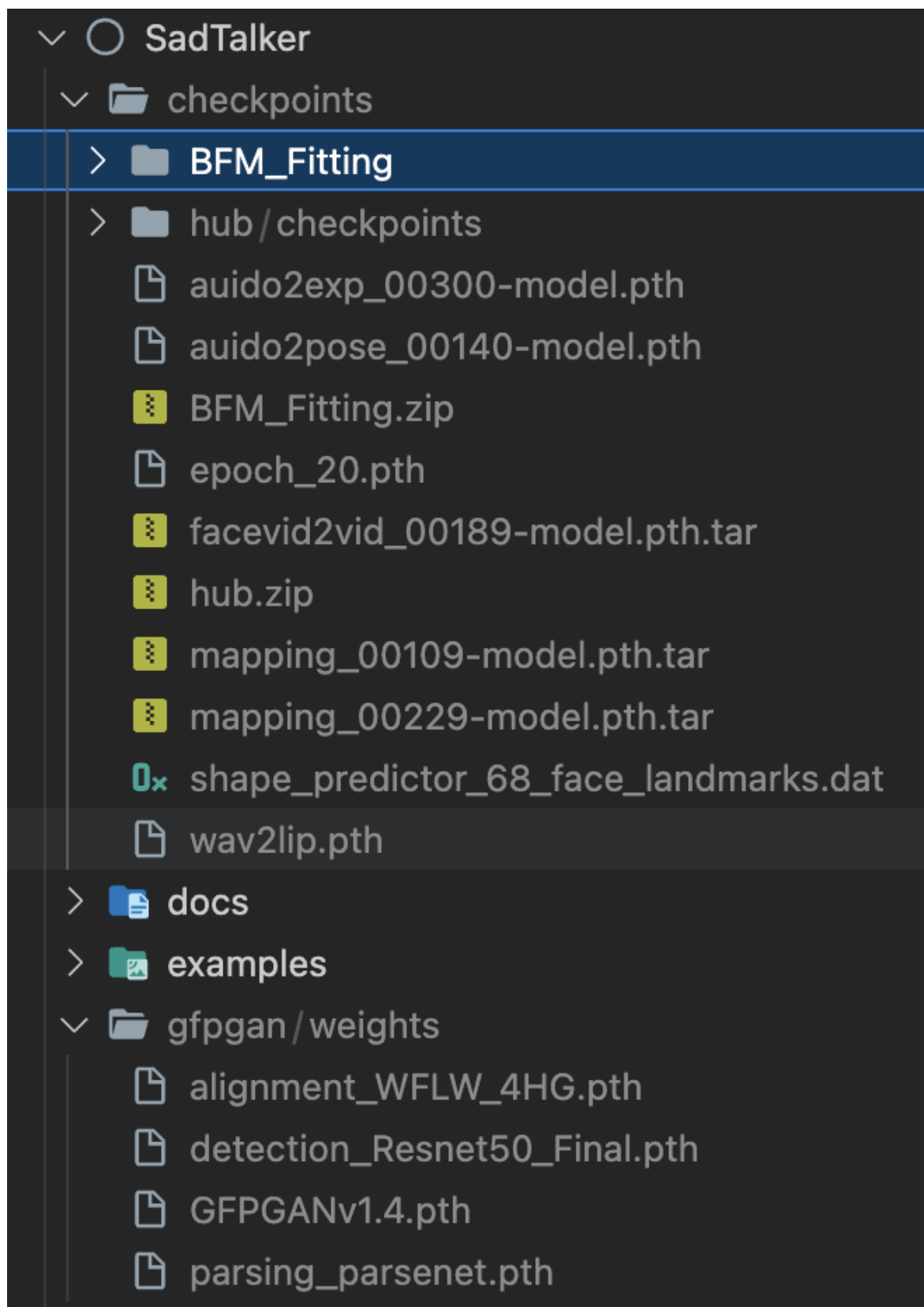
还会下载其它文件，建议手工下载，复制到对应目录中：

Downloading:

["https://github.com/xinntao/faceXlib/releases/download/v0.1.0/alignment_WFLW_4HG.pth"](https://github.com/xinntao/faceXlib/releases/download/v0.1.0/alignment_WFLW_4HG.pth)

to D:\ai\stable-diffusion-

webui\extensions\SadTalker\gfpgan\weights\alignment_WFLW_4HG.pth



sd-webui-and-easyphoto

easyphoto

[] 已实现

手动 `git clone git@github.com:aigc-apps/sd-webui-EasyPhoto.git` 到本地，再复制到 `D:\ai\stable-diffusion-webui\extensions` 中。

在sd-web-ui的extensions标签页的 `Install from URL` 中填 `git@github.com:aigc-apps/sd-webui-EasyPhoto.git`，再点Install。再去 `Installed` 标签页点 `Apply and restart UI`。

启动时报错：

```
*** Error loading script: easyphoto_ui.py
Traceback (most recent call last):
  File "D:\ai\stable-diffusion-webui\venv\lib\site-packages\modelscope\utils\ast_utils.py", line 466, in _
    output = self.astScanner.generate_ast(file)
  File "D:\ai\stable-diffusion-webui\venv\lib\site-packages\modelscope\utils\ast_utils.py", line 365, in g
    output = self.scan_import(node, show_offsets=False)
  File "D:\ai\stable-diffusion-webui\venv\lib\site-packages\modelscope\utils\ast_utils.py", line 164, in s
    local_out = _scan_import(el, type(el).__name__)
  File "D:\ai\stable-diffusion-webui\venv\lib\site-packages\modelscope\utils\ast_utils.py", line 133, in _
    return self.scan_import(
  File "D:\ai\stable-diffusion-webui\venv\lib\site-packages\modelscope\utils\ast_utils.py", line 151, in s
    attr = getattr(node, field)
AttributeError: 'ClassDef' object has no attribute 'type_params'
```

删除 `~/cache/modelscope/ast_indexer` 这个文件，再重启sd-webui即可。

```
*** Error loading script: easyphoto_ui.py
Traceback (most recent call last):
  File "D:\ai\stable-diffusion-webui\modules\scripts.py", line 515, in load_scripts
    script_module = script_loading.load_module(scriptfile.path)
  File "D:\ai\stable-diffusion-webui\modules\script_loading.py", line 13, in load_module
    module_spec.loader.exec_module(module)
  File "<frozen importlib._bootstrap_external>", line 883, in exec_module
  File "<frozen importlib._bootstrap>", line 241, in _call_with_frames_removed
  File "D:\ai\stable-diffusion-webui\extensions\sd-webui-EasyPhoto\scripts\easyphoto_ui.py", line 23, in <
    from scripts.easyphoto_train import easyphoto_train_forward
  File "D:\ai\stable-diffusion-webui\extensions\sd-webui-EasyPhoto\scripts\easyphoto_train.py", line 23, i
    from scripts.train_kohya.utils.lora_utils import convert_lora_to_safetensors
  File "D:\ai\stable-diffusion-webui\extensions\sd-webui-EasyPhoto\scripts\train_kohya\utils\lora_utils.py
    from diffusers import AutoencoderKL
  File "D:\ai\stable-diffusion-webui\venv\lib\site-packages\diffusers\_init_.py", line 5, in <module>
    from .utils import (
  File "D:\ai\stable-diffusion-webui\venv\lib\site-packages\diffusers\utils\_init_.py", line 38, in <mod
    from .dynamic_modules_utils import get_class_from_dynamic_module
  File "D:\ai\stable-diffusion-webui\venv\lib\site-packages\diffusers\utils\dynamic_modules_utils.py", lin
    from huggingface_hub import HfFolder, cached_download, hf_hub_download, model_info
```

```
ImportError: cannot import name 'cached_download' from 'huggingface_hub' (D:\ai\stable-diffusion-webui\ven
```

本机解决方案见[这里](#)。【

SOLUTION : Replace `cached_download` by `hf_hub_download` in `D:\ai\stable-diffusion-webui\venv\lib\site-packages\diffusers\utils\dynamic_modules_utils.py`】

生成时报错：

```
Applying attention optimization: InvokeAI... done.
Traceback (most recent call last):
  File "D:\ai\stable-diffusion-webui\venv\lib\site-packages\gradio\routes.py", line 488, in run_predict
    output = await app.get_blocks().process_api(
  File "D:\ai\stable-diffusion-webui\venv\lib\site-packages\gradio\blocks.py", line 1431, in process_api
    result = await self.call_function(
  File "D:\ai\stable-diffusion-webui\venv\lib\site-packages\gradio\blocks.py", line 1103, in call_function
    prediction = await anyio.to_thread.run_sync(
  File "D:\ai\stable-diffusion-webui\venv\lib\site-packages\anyio\to_thread.py", line 33, in run_sync
    return await get_asynclib().run_sync_in_worker_thread(
  File "D:\ai\stable-diffusion-webui\venv\lib\site-packages\anyio\backends\asyncio.py", line 877, in run_sync
    return await future
  File "D:\ai\stable-diffusion-webui\venv\lib\site-packages\anyio\backends\asyncio.py", line 807, in run
    result = context.run(func, *args)
  File "D:\ai\stable-diffusion-webui\venv\lib\site-packages\gradio\utils.py", line 707, in wrapper
    response = f(*args, **kwargs)
  File "D:\ws\apps\python310\lib\contextlib.py", line 79, in inner
    return func(*args, **kwargs)
  File "D:\ai\stable-diffusion-webui\extensions\sd-webui-EasyPhoto\scripts\easyphoto_train.py", line 228, in e
    unload_models()
  File "D:\ai\stable-diffusion-webui\extensions\sd-webui-EasyPhoto\scripts\easyphoto_utils\common_utils.py", l
    torch.cuda.ipc_collect()
  File "D:\ai\stable-diffusion-webui\venv\lib\site-packages\torch\cuda\_init_.py", line 795, in ipc_collect
    _lazy_init()
  File "D:\ai\stable-diffusion-webui\venv\lib\site-packages\torch\cuda\_init_.py", line 298, in _lazy_init
    torch._C._cuda_init()
RuntimeError: Found no NVIDIA driver on your system. Please check that you have an NVIDIA GPU and installed a
```

解决方式在[这里](#)。

修改 `D:\ai\stable-diffusion-webui\extensions\sd-webui-`

`EasyPhoto\scripts\easyphoto_utils\common_utils.py` 的 816 行及下一行：

```
torch.cuda.empty_cache()
torch.cuda.ipc_collect()
```

为

```
if torch.cuda.is_available():
    torch.cuda.empty_cache()
    torch.cuda.ipc_collect()
```

并删除同目录下的 `__pycache__` 文件夹

重新生成或重启sd-webui后再重新生成。

报错：`RuntimeError: "slow_conv2d_cpu" not implemented for 'Half'`
暂未解决，暂时不使用easyphoto了。

sd-webui-cleaner

<https://github.com/novitalabs/sd-webui-cleaner>

手动 `git clone git@github.com:novitalabs/sd-webui-cleaner.git` 到本地，再复制到 D:\ai\stable-diffusion-webui\extensions 中。

去除水印

- 用法：<https://www.bilibili.com/video/BV1coUGYPE7J/>

若报错

```
File "/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/gradio/components/gallery.py", line 100, in process_image
    raise ValueError(f"Cannot process type as image: {type(img)}")
ValueError: Cannot process type as image: <class 'NoneType'>
```

就依次点击 `Settings` ，搜索 `Uncategorized` ，它下面有 `Cleaner` ，不勾选 `Is Use GPU` ，再重启即可。

sd-webui-text2video

<https://github.com/kabachuha/sd-webui-text2video>

手动 `git clone git@github.com:kabachuha/sd-webui-text2video.git` 到本地，再复制到 D:\ai\stable-diffusion-webui\extensions中。

下载[模型](#)，放到stable-diffusion-webui/models/ModelScope/t2v中。

sd-webui-animatediff

<https://github.com/continue-revolution/sd-webui-animatediff>

手动 `git clone git@github.com:continue-revolution/sd-webui-animatediff.git` 到本地，再复制到D:\ai\stable-diffusion-webui\extensions中。

下载模型，放到stable-diffusion-webui/extensions/sd-webui-animatediff/model中。

sd-webui-segment-anything

tts

tts

tts-stt-by-buzz

keywords: `tts` `stt` `buzz`

buzz

download and install

[buzz-github](#)

go to [realase page](#) to download `Buzz-Windows-X64.zip` . the version I'm using is `v1.2.0` .

unpack the zip, and double click .exe file.

usage

run Buzz and import an audio file (like mp3), choose model(Buzz will download model file automatically)

<https://chidiwilliams.github.io/buzz/docs/faq>

official doc

<https://chidiwilliams.github.io/buzz/docs>

commandline

Translate two MP3 files from French to English using OpenAI Whisper API

```
buzz add --task translate --language fr --model-type openaiapi  
/Users/user/Downloads/1b3b03e4-8db5-ea2c-ace5-b71ff32e3304.mp3  
/Users/user/Downloads/koaf9083k1lkpsfdi0.mp3
```

Transcribe an MP4 using Whisper.cpp "small" model and immediately export to SRT and VTT files

```
buzz add --task transcribe --model-type whispercpp --model-size small --prompt "My initial  
prompt" --srt --vtt /Users/user/Downloads/buzz/1b3b03e4-8db5-ea2c-ace5-  
b71ff32e3304.mp4
```

edge-tts

install

```
pip install edge-tts
```

using by command-li

```
edge-tts --help
edge-tts --text "Hello, world!" --write-media hello.mp3 --write-subtitles hello.vtt
edge-tts --list-voices
Name: zh-CN-XiaoxiaoNeural Female
Name: zh-CN-XiaoyiNeural Female
Name: zh-CN-YunjianNeural Male
Name: zh-CN-YunxiNeural Male
Name: zh-CN-YunxiaNeural Male
Name: zh-CN-YunyangNeural Male
Name: zh-CN-liaoning-XiaobeiNeural Female
Name: zh-CN-shaanxi-XiaoniNeural Female
Name: zh-HK-HiuGaiNeural Female
Name: zh-HK-HiuMaanNeural Female
Name: zh-HK-WanLungNeural Male
Name: zh-TW-HsiaoChenNeural Female
Name: zh-TW-HsiaoYuNeural Female
Name: zh-TW-YunJheNeural Male

edge-tts --rate=+1% --volume=+1% --pitch=+5Hz --text "Hello, world!" --write-media hello.mp3 --write-subtitles hello.vtt --voice zh-CN-XiaoxiaoNeural

edge-tts --rate=+1% --volume=+1% --pitch=+5Hz --text "静夜思 - 李白 床前明月光，疑是地上霜。举头望明月，低头思故乡。" --write-media jys.mp3 --write-subtitles jys.vtt --voice zh-CN-XiaoxiaoNeural

edge-tts --rate=+1% --volume=+1% --pitch=+5Hz --file "静夜思-李白.txt" --write-media jys.mp3 --write-subtitles jys.vtt --voice zh-CN-XiaoxiaoNeural
```

using by python code

```
#!/usr/bin/env python3
# -*- coding: utf8 -*-
import subprocess
import uuid

import edge_tts
import asyncio

#####
# pip3 install edge_tts -i https://mirrors.aliyun.com/pypi/simple/
#####

async def text_to_mp3(text, mp3_name):
    tts = edge_tts.Communicate(text=text, voice='zh-CN-XiaoxiaoNeural', rate='+1%', volume='+1%', pitch='+5Hz')
    await tts.save(str(mp3_name))

if __name__ == '__main__':
    text_to_use = 'Stable Diffusion WebUI, 简称SD WebUI, 是一个基于Gradio库的Stable Diffusion的浏览器界面。它支持目前主流的开源AI绘画模型, 如NovelAI/Stable Diffusion等。通过SD WebUI, 用户可以方便地配置和生成AI绘画作品, 实现精细化的创作。'
    asyncio.run(text_to_mp3(text_to_use, "a.mp3"))
```

附件

静夜思-李白.txt

```
静夜思 - 李白
床前明月光，疑是地上霜。
举头望明月，低头思故乡。
```

tts链接

文本转语音 tts (Text To Speech)

- <https://ttsmaker.com/zh-cn>
- <https://app.xunjiepdf.com/text2voice/>
- <https://github.com/LokerL/tts-vue/releases>

语音转文本 stt

- <https://app.xunjiepdf.com/voice2text/>
- pc软件：
 - [WhisperDesktop](#)
 - [Buzz](#)

ai常用链接

关键字

ai常用链接

常用链接

- [图片去水印 - 在线处理](#)
- [免费在线去水印工具，快速视频去水印图片水印](#)
- [免费在线转换HEIC至JPG格式。支持 **批量**](#)
- [腾讯元宝 - 轻松工作 多点生活](#)
- [一文了解：目前所有常用AI大盘点，你用过哪些？ - 知乎](#)
- [Beautiful Free Images & Pictures | Unsplash](#)
- [ai生成短剧](#)
- [AI工具集官网 | 1000+ AI工具集合，国内外AI工具集导航大全](#)

ai写作工具

- ai文字内容创作工具：[AI that's built for marketing – Jasper](#)
- ai营销方案和ai内容创作工具：[Future proof your business with GTM AI](#)
- writesonic: AI写作、文案、释义工具
- Article Forge：AI文章生成器
- Ink for All：具有必定、写作和生成SEO功能的工具
- AI Writer：AI内容生成平台

AI图像工具

- Midjourney
- bing image creator
- DeepArt和DeepDream Generator

AI音频工具

- 网易天音：AI音乐创作平台

- Riffusion : 生成不同风格音乐的工具
- 讯飞智作 : 文字转语音和配音工具

AI视频工具

- Runway : 视频生成、动态捕捉
- D-ID : AI真人口播视频生成工具
- Artflow : AI生成视频动画
- Synthesia : AI视频制作工具

AI设计工具

- Microsoft Designer
- Magic Design

AI代码生成工具

- Amazon CodeWhisperer
- github copilot
- Codeium

AI编程

codex充值 : <https://bewild.ai/subscribe>
<https://getgpt.pro/>

windsurf (idea plugin, vs code plugin)

如果 Codex / opencode / curl / git 访问外网失败，可搜索“Bash 中配置代理函数”寻找解决方案。

ai code generater

free

Amazon CodeWhisperer

non-free

github copilot

opencode

info

official website: [OpenCode | The open source AI coding agent](#)

installation

you should open magic tool to install opencode.

cli

windows

1. install bun

```
powershell -c "irm bun.sh/install.ps1|iex"
```

执行失败就手工下载 <https://github.com/oven-sh/bun/releases/latest/download/bun-windows-x64.zip> 并放到C:\Users\Administrator.bun\bin\bun-windows-x64.zip中

2. open new cmd, and

```
bun add -g opencode-ai
```

linux / wsl

suggestion is via node(v20.x).

vs code

in vscode, install extension 'opencode'.

click 'Open opencode in new tab' and you will see opencode tool.

ide

no official plugins for jetbrains idea.
suggestion is using terminal (and typing opencode).

plugins

vim "C:\Users\Administrator.config\opencode\opencode.json"

```
{
  "$schema": "https://opencode.ai/config.json",
  "plugin": [
    "superpowers@git+https://github.com/obra/superpowers.git",
    "oh-my-openagent@latest"
  ]
}
```

superpower

<https://github.com/obra/superpowers>

<https://raw.githubusercontent.com/obra/superpowers/refs/heads/main/.opencode/INSTALL.md>

oh-my-opencode

website: <https://github.com/opensoft/oh-my-opencode>

```
bunx oh-my-opencode install
```

files

```
C:\Users\Administrator.config\opencode
| ----- opencode.json
| ----- oh-my-opencode.json
```

filename oh-my-opencode.json

<https://github.com/ziho7/opencode-config/blob/main/oh-my-opencode.json>

```
{
  "$schema": "https://raw.githubusercontent.com/code-yeongyu/oh-my-opencode/master/assets/oh-my-opencode.schema.json",
  "agents": {
```

```
"librarian": {
  "model": "opencode/glm-4.7-free"
},
"explore": {
  "model": "opencode/grok-code"
},
"oracle": {
  "model": "opencode/grok-code"
},
"frontend-ui-ux-engineer": {
  "model": "gemini/gemini-3-pro-preview"
},
"document-writer": {
  "model": "gemini/gemini-3-flash"
},
"multimodal-looker": {
  "model": "gemini/gemini-3-flash"
}
},
"categories": {
  "quick": {
    "model": "opencode/glm-4.7-free"
  },
  "deep": {
    "model": "opencode/grok-code"
  },
  "ultrabrain": {
    "model": "opencode/grok-code"
  },
  "writing": {
    "model": "gemini/gemini-3-flash"
  },
  "visual-engineering": {
    "model": "gemini/gemini-3-pro-preview"
  },
  "artistry": {
    "model": "gemini/gemini-3-pro-preview"
  },
  "unspecified-low": {
    "model": "opencode/glm-4.7-free"
  },
  "unspecified-high": {
    "model": "opencode/grok-code"
  }
}
}
```


claude

installation

Suggestion is using <https://github.com/anthropics/claude-code/releases>. Downloading it and put it in PATH.

```
you can put it in ~/.local/bin and append ~/.local/bin to $PATH
```

using minimax model

create file ~/.claude/.claude.json

```
{
  "hasCompletedOnboarding": true
}
```

create file ~/.claude/settings.json

see <https://platform.minimaxi.com/docs/guides/text-ai-coding-tools>

```
{
  "env": {
    "ANTHROPIC_BASE_URL": "https://api.minimaxi.com/anthropic",
    "ANTHROPIC_AUTH_TOKEN": "sk-cp-xxxx",
    "API_TIMEOUT_MS": "3000000",
    "CLAUDE_CODE_DISABLE_NONESSENTIAL_TRAFFIC": "1",
    "ANTHROPIC_MODEL": "MiniMax-M2.7",
    "ANTHROPIC_SMALL_FAST_MODEL": "MiniMax-M2.7",
    "ANTHROPIC_DEFAULT_SONNET_MODEL": "MiniMax-M2.7",
    "ANTHROPIC_DEFAULT_OPUS_MODEL": "MiniMax-M2.7",
    "ANTHROPIC_DEFAULT_HAIKU_MODEL": "MiniMax-M2.7"
  },
  "enabledPlugins": {
    "gopls-lsp@claude-plugins-official": true
  },
  "theme": "light-daltonized"
}
```

多媒体工具

FFmpeg 音频视频处理

关键字：`ffmpeg` `音视频` `音频视频`

截取

提取视频中的音频 (-vn表示no video, -c:a 是codec of audio的意思, copy是直接拷贝视频中的原始的音频, 这里不会涉及音频的编解码, 速度会很快。)

```
ffmpeg -i input.mp4 -vn -c:a copy output.aac
```

导出mp3格式的音频 (-c:a mp3)

```
ffmpeg -i input.mp4 -vn -c:a mp3 output.mp3
```

删除视频中的音频: (-an表示no audio)

```
ffmpeg -i input.mp4 -an -c:v copy output.mp4
```

从第00:01秒开始截取到最后

```
ffmpeg -ss 00:01 -i input.mp4 -c:v copy -c:a copy output.mp4
```

从第00:01秒开始截取31秒

```
ffmpeg -ss 00:01 -i input.mp4 -t 31 -c:v copy -c:a copy output.mp4
```

从第00:00秒开始截取到14:33

```
ffmpeg -ss 00:00 -i output.mp4 -to 14:33 -c:v copy -c:a copy output2.mp4
```

从第00:01秒开始截取到最后

```
ffmpeg -ss 00:00:01 -i input.mp3 -c:a copy output.mp3
```

批处理

```
for /r %i in (*.mov) do ffmpeg -i %i -an -c:v copy %i.mp4
```

合并

合并ts

```
ffmpeg -i "concat:0001.ts|0002.ts|0003.ts" -c copy output.mp4
```

合并mp4

现有a1.mp4,a2.mp4,a3.mp4,想合并成一个a_all.mp4。

创建source_list.txt,内容是

```
file 'a1.mp4'  
file 'a2.mp4'  
file 'a3.mp4'
```

然后使用 `ffmpeg -f concat -i source_list.txt -c copy a_all.mp4`

合并字幕和mp4

merge ass and mp4:

```
ffmpeg -i input.mp4 -vf ass=abcd.ass -f mp4 output.mp4
```

```
ffmpeg -i input.mp4 -vf ass=abcd.ass -f mp4 -vcodec libx264 -crf 2 -y output.mp4
```

合并音频与视频：

不指定时间(即在第1秒)

```
ffmpeg.exe -i p1.mp4 -i sound.ogg -vcodec copy -acodec copy p1_with_sound.mp4
```

指定时间

```
ffmpeg -i p1.mp4 -i sound.ogg -filter_complex "[1]adelay=5000|5000[a]" -map 0:v:0 -map "[a]" -c:v copy -c:a aa
```

上面的命令是，不改变p1.mp4时把sound.ogg插入到第5秒（5000毫秒）处播放。

把一个新音频合并到视频（已有音频）中

```
ffmpeg -i video.mp4 -i new_audio.mp3 -filter_complex "[1:a]adelay=5000|5000[newaud];[0:a][newaud]amix=inputs=2
```

个人理解：`[1:a]adelay=5000|5000[newaud]` 是将第2个音频延迟5秒播放并命名为newaud，`[0:a][newaud]amix=inputs=2[newmix]` 是第一个视频的音频和newaud合并成一个，并命名为newmix，`-map "[newmix]"` 是使用newmix作为输出。

添加水印

图片水印

Adding Watermark at the Bottom-Right Corner

```
ffmpeg -i p1.mp4 -i watermark.png -filter_complex "overlay=W-w-10:H-h-10" -c:v copy -c:a copy -y p1-with-watermark-image.mp4
```

this command will decrease video frame rates.

Explanation:

W and H represent the width and height of the video, respectively.
 w and h represent the width and height of the watermark image, respectively.
 W-w-10 positions the watermark 10 pixels from the right edge.
 H-h-10 positions the watermark 10 pixels from the bottom edge.

You can find the original frame rate of the video by running:
 Look for the `fps` value in the output, find fps

```
ffmpeg -i p1.mp4
```

```
-c:v libx264 -r 30
```

```
# command:
```

```
ffmpeg -i p1.mp4 -i watermark.png -filter_complex "overlay=x=W-w-10:y=H-h-10" -c:a copy -c:v libx264 -crf 2 -
```

```
# example:
```

```
ffmpeg -i p1.mp4 -i watermark.png -filter_complex "overlay=x=W-w-10:y=H-h-10" -c:a copy -c:v libx264 -crf 2
```

video quality is higher when crf is small.

指定时间显示水印：

下面的命令在第2秒~第10秒显示水印

```
ffmpeg -i p1.mp4 -stream_loop -1 -i click.gif -filter_complex "[0:v][1:v]
overlay=x=W-w-90:y=H-h-30:enable='between(t,2,10)':shortest=1" -c:a copy -c:v
libx264 -crf 2 -b:v 793k -r 30 -y p1-with-gif-with-duration3.mp4
```

添加文字水印

```
# loss fps
```

```
ffmpeg -i p1.mp4 -filter_complex "drawtext=text='Watermark':fontfile=msyh.ttc:x=W-text_w-10:y=10:fontsize=36:f
```

```
# more higher quality
```

```
ffmpeg -i p1.mp4 -filter_complex "drawtext=text='Watermark':fontfile=msyh.ttc:x=W-text_w-10:y=10:fontsize=36:f
`box=1` will give watermark text a background.
```

```
# 横向跑马灯效果
```

```
ffmpeg -i p1.mp4 -filter_complex "drawtext=text='Watermark':fontfile=msyh.ttc:x='if(gte(t,1),mod(t*100, W+tw)-
`x='if(gte(t,1),mod(t*100, W+tw)-tw, NAN)': This expression moves the text horizontally across the screen star
```

Copy 微软雅黑字体(msyh.ttc) to current directory.

I use c:/windows/font/msyh.ttc, but it does not work.

转换

aac转mp3

```
ffmpeg -i input.aac -acodec libmp3lame input.mp3
```

音频处理

音频拼接

```
ffmpeg -i "concat:test.mp3|radio.mp3" -acodec copy output.mp3
```

将多个MP3按顺序拼接生成一个mp3

补白

手动生成一条10秒长的空白音频

```
ffmpeg -f lavfi -t 10 -i anullsrc test.mp3 -y
```

添加转场

```
ffmpeg -i p1.mp4 -i p2.mp4 -filter_complex  
xfade=transition=rectcrop:duration=2s:offset=8s -y output_3.mp4
```

```
ffmpeg -i 1.mp4 -i 2.mp4 -filter_complex xfade=transition=rectcrop:duration=1:offset=5 -y  
3.mp4
```

duration 持续时间（设置交叉淡出持续时间，以秒为单位。默认为1秒）

offset 偏移量（设置交叉淡入开始相对于第一个输入流的秒。默认偏移量为0）

transition 过渡效果

custom | fade | wipeleft | wiperight | wipeup | wipedown | slideleft | slideright | slideup |
slidedown | circlecrop | rectcrop | distance | fadeblack | fadewhite | radial | smoothleft |
smoothright | smoothup | smoothdown | circleopen | circleclose | vertopen | vertclose | horzopen
| horzclose | dissolve | pixelize | diagtl | diagtr | diagbl | diagbr | hlslice | hrslice | vuslice | vdslice
| hblur | fadegrays | wipetl | wipetr | wipebl | wipebr | squeezeh | squeezev | zoomin

图片转视频

单张图片转视频

```
ffmpeg -r 25 -f image2 -loop 1 -i input2.jpg -vcodec libx264 -pix_fmt yuv420p -s 732x1200 -r 25 -t 10 -crf 2 -y output_3.png.mp4
```

这里需要注意-r 25 参数的位置，在-i input2.jpg前面和在后面的效果是不一样的。放在-i后面只会改变输出的视频帧率，而输入的还是默认值25。所以需要两次

多张图片转视频

```
ffmpeg -framerate 0.33 -f image2 -loop 1 -i images/a_%d.jpeg -vcodec libx264 -pix_fmt yuv420p -s 1080*1920 -r 25 -t 30 -y output_images2mp4_1.mp4
```

```
-r 25 帧率，表示每秒播放帧数，默认帧率为25，也就是1秒钟拼接25张图片，我们可以通过调整帧率的大小来控制最终生成
-framerate 0.33 每秒播放图片张数，算法：round(1 / (视频总时长 / 图片总张数)， 2)
-f image2 输入流文件格式
-loop 1 输入流循环次数，仅对图片有效，0表示无限循环
-i fps_%d.jpg 文件名，%d、%2d表示匹配数字序列
-vcodec libx264 视频编码，缺少时h5中可能无法播放
-pix_fmt yuv420p 视频格式，缺少时h5中可能无法播放
-s 1080*1920 视频分辨率，生成的视频最终尺寸
-t 30 视频总时长，以秒为单位
-y fps.mp4 覆盖视频文件
```

视频画中画

<https://www.nxrte.com/jishu/8456.html>

```
# 1. 创建主画面视频素材
ffmpeg -f lavfi -i testsrc2=s=800x480 -vcodec libx264 -t 10 input1.mp4
# 2. 创建子画面素材
ffmpeg -f lavfi -i testsrc2=s=320x240 -vcodec libx264 -t 5 input2.mp4
# 3. 开始画中画操作
ffmpeg -i input1.mp4 -filter_complex "movie=input2.mp4[in2];[in2]setpts=PTS+2/TB[out2];[0:v][out2]overlay=x=2
# 我们解析一下命令行参数：
# 输入的主文件是 input1.mp4
# 输入的子文件是 input2.mp4
# 将子画面的所有的时间戳都加上 2 秒钟的偏移
# 将子画面显示在主画面的 x 坐标为 20, y 坐标为 120 的位置
# 子画面在主画面显示时间段是第 2 秒到第 4 秒之间
# 以主画面和子画面中时长最短的那个视频流长度为结束点 (input1.mp4是10秒, input2.mp4是5秒, picture_in_pic.mp4是s
```

```
# 生成的视频有声音，是第二个视频的
ffmpeg -i input1.mp4 -i overlay.mp4 -filter_complex "[0:v][1:v]overlay=x=W-w-10:y=H-h-30,format=yuv420p[out]"
```