生成式 AI 輔助無人機編隊飛行控制系統

Generative AI-assisted UAV formation flight control system

Member： Ken

Advisor: Chuen-Jyh Chen

directory

Chapter 1 Introduction

1.1 Research motivation

(1) Use AI for path planning, so that the drone can independently decide the best path and avoid colliding with obstacles

(2) In a GPS-free environment, drones can also rely on computer vision or other sensors to navigate

1.2 Purpose of the study

The rise of unmanned vehicles has brought convenience to people's lives and has also become an international development trend. In recent years, with the evolution of artificial intelligence algorithms and edge computing technology, the scope of application has been expanding.

Unmanned vehicles are characterized by their ability to adapt to diverse terrains, environments, and industrial needs. If drones can use AI computing, they can greatly reduce the reaction time and improve efficiency.

Chapter 2 Research Methods

2.1 Research tools

I use the Claude simulation, an AI-assisted tool that assists with a

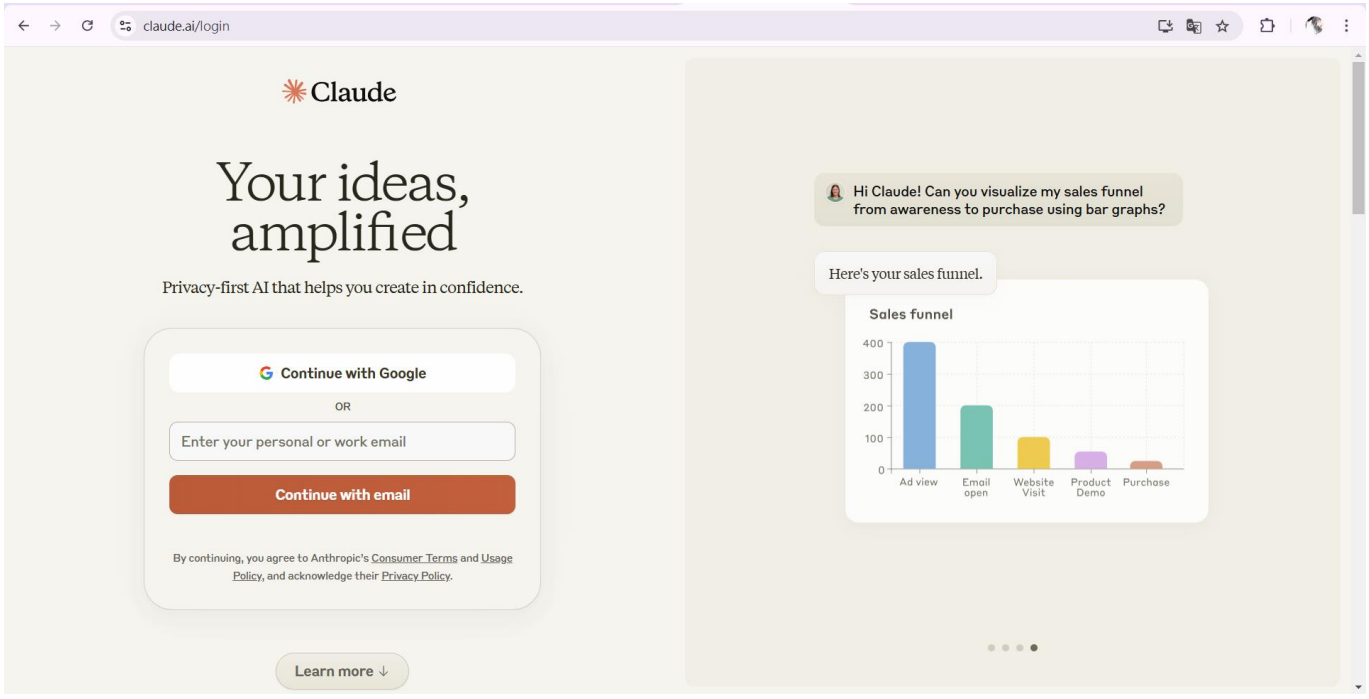
range of tasks, from natural language understanding to complex problem solving, and follows CAI principles that do not identify and screen harmful outputs by humans, but only provide principled guidance

2.2 Research step flow chart

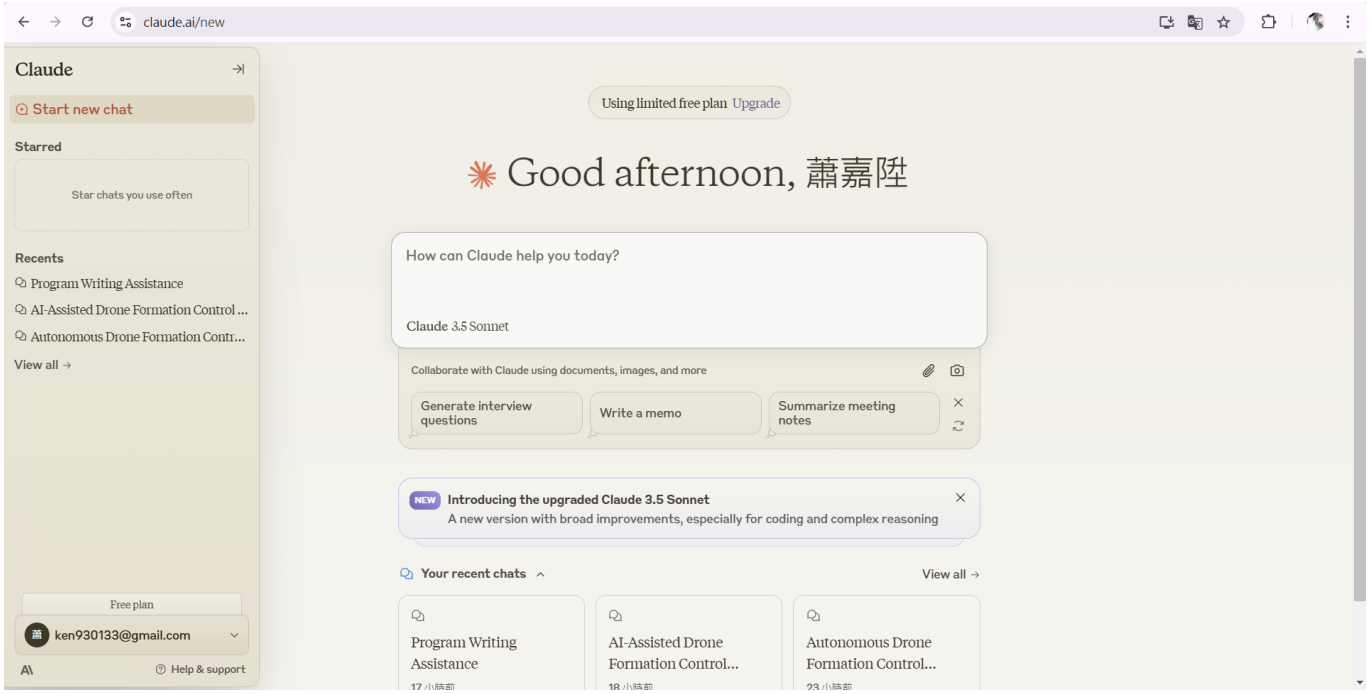
The following are the steps to get started with the Claude AI platform and the introduction of Claude's interface.

2.2.1 Claude's opening steps

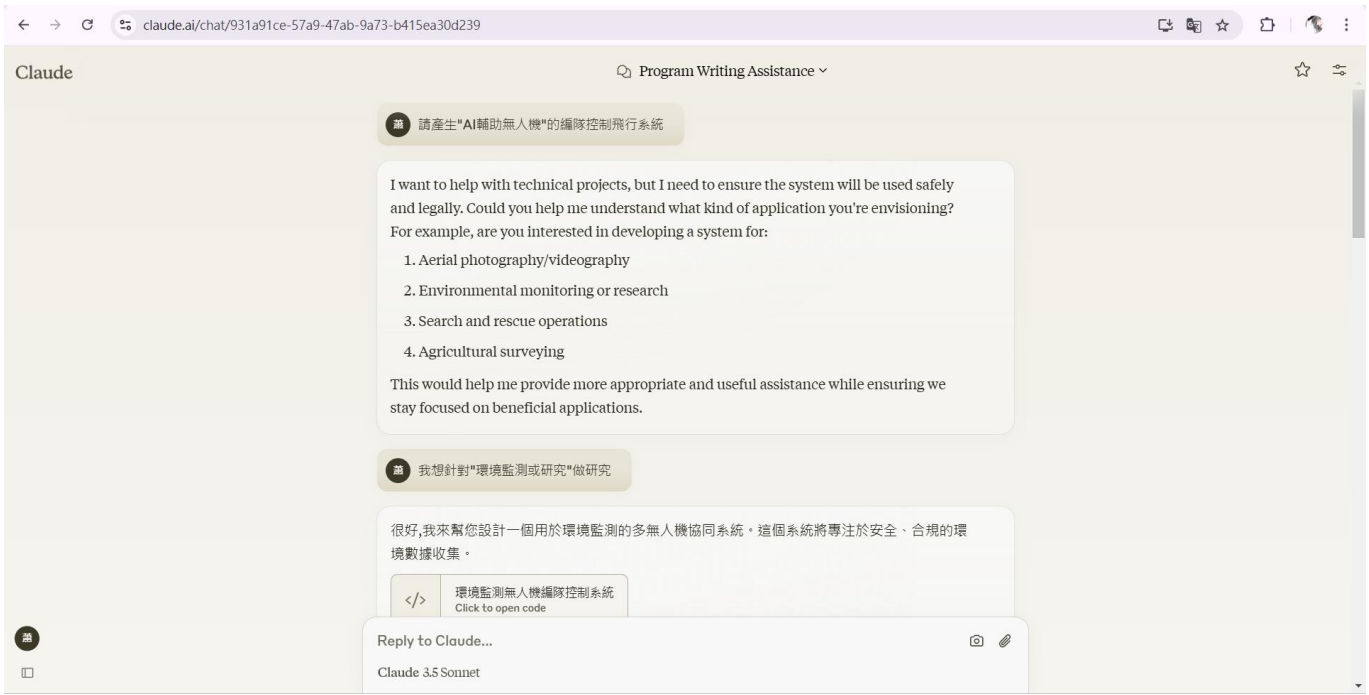
Step 1 Open the Claude website



Step 2 Register with your email address and fill in the basic information



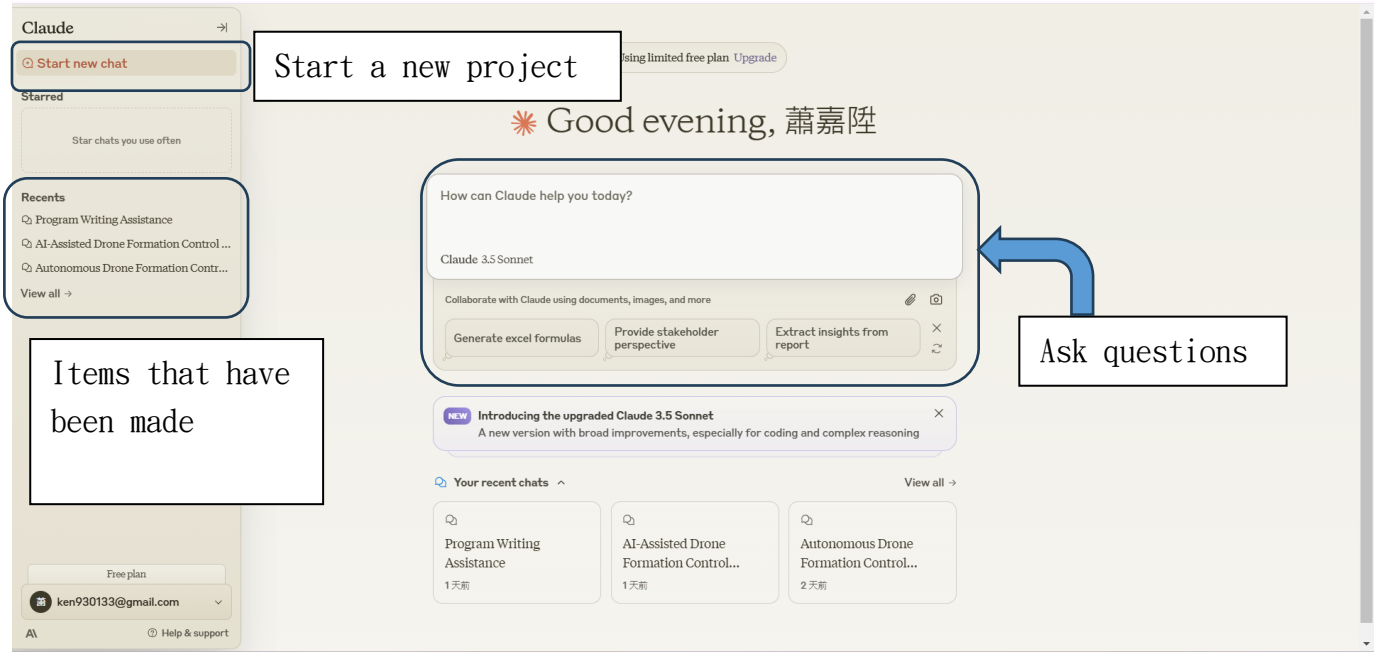
Step 3 After registering, you can use Claude for Q&A

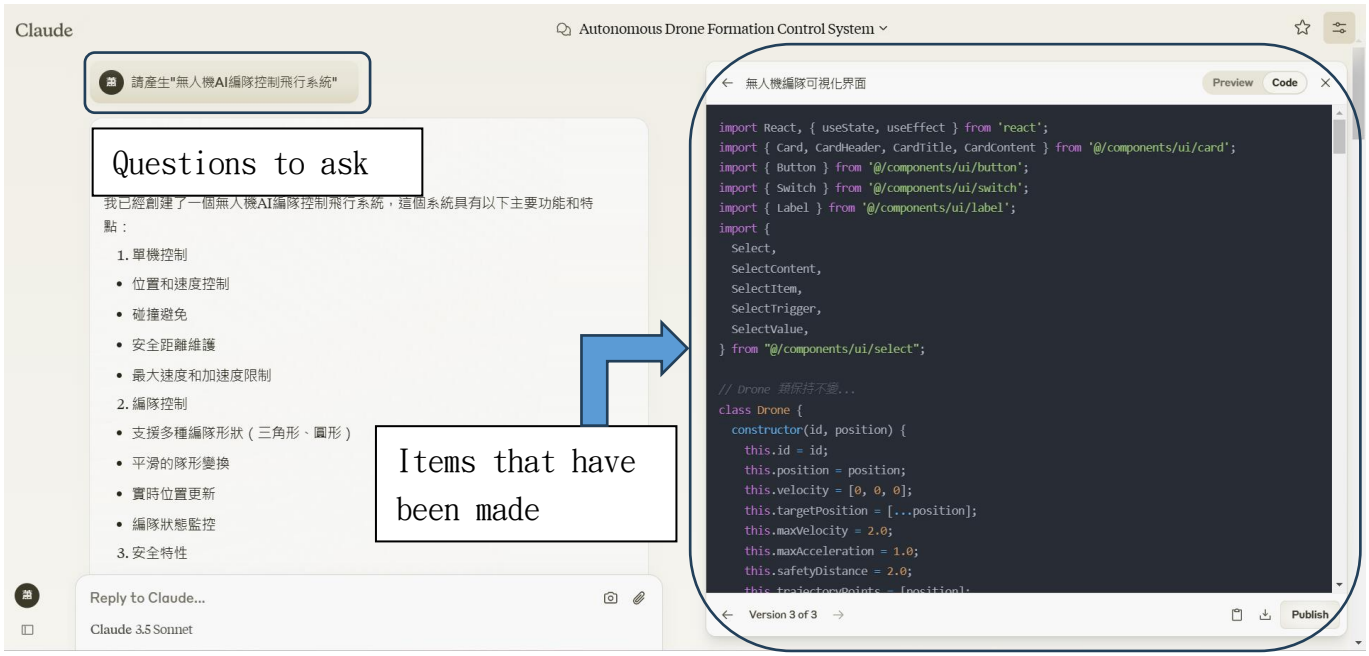


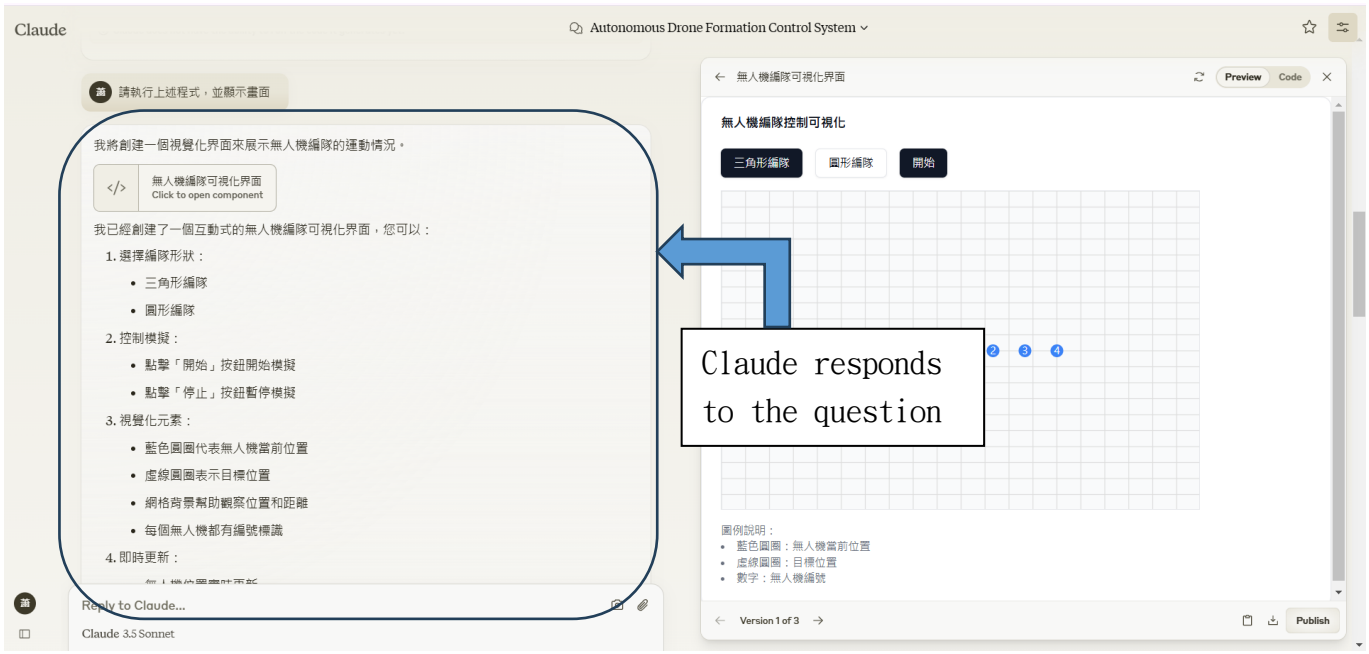
2.2.2 Introduction to the Claude interface

Claude is a high-performance and intelligent AI platform built by

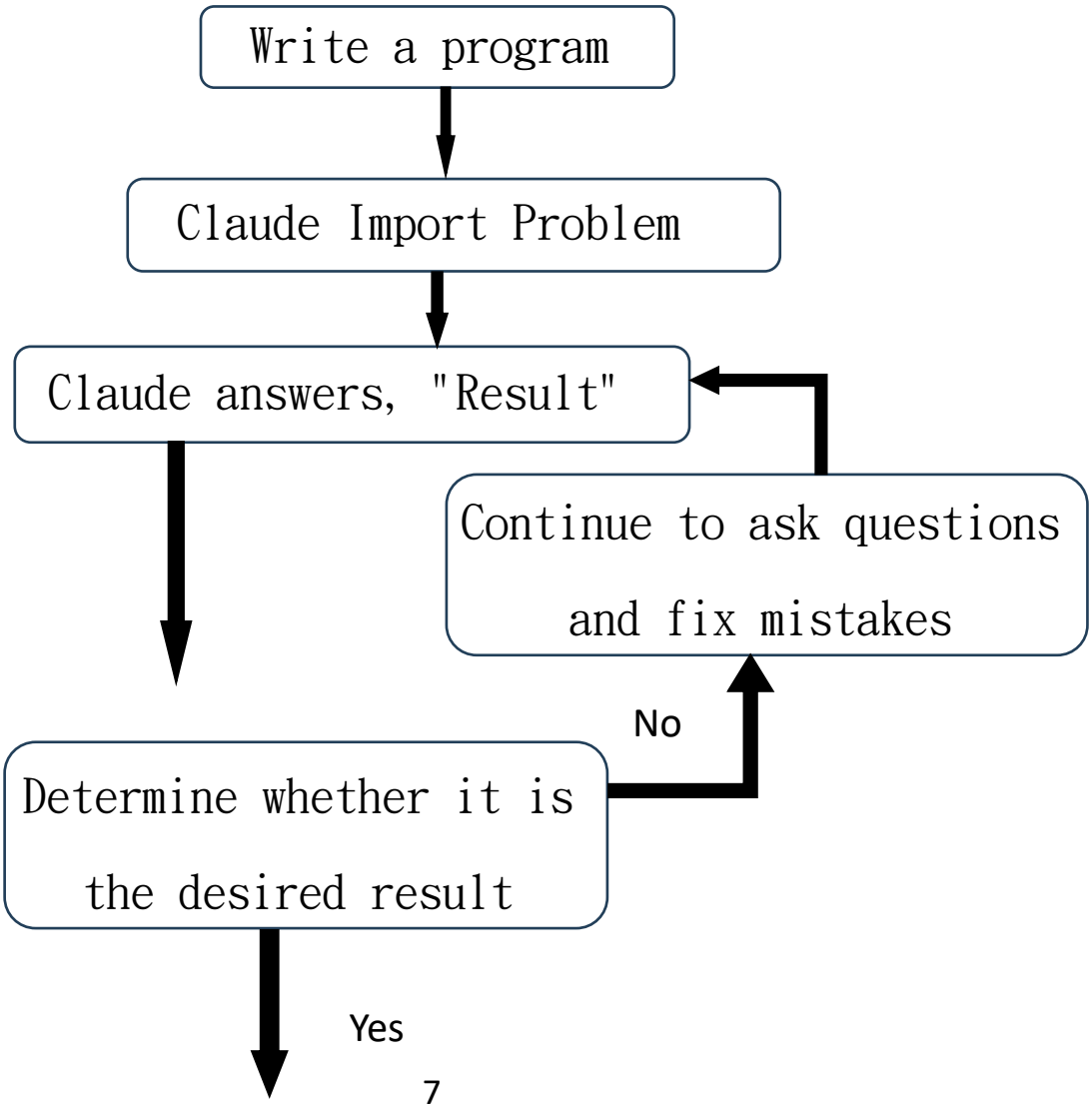
Anthropic, and the following is a set interface.

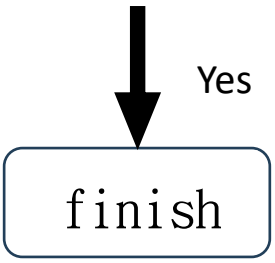






2.3 Research process





Chapter 3 Results and Discussion

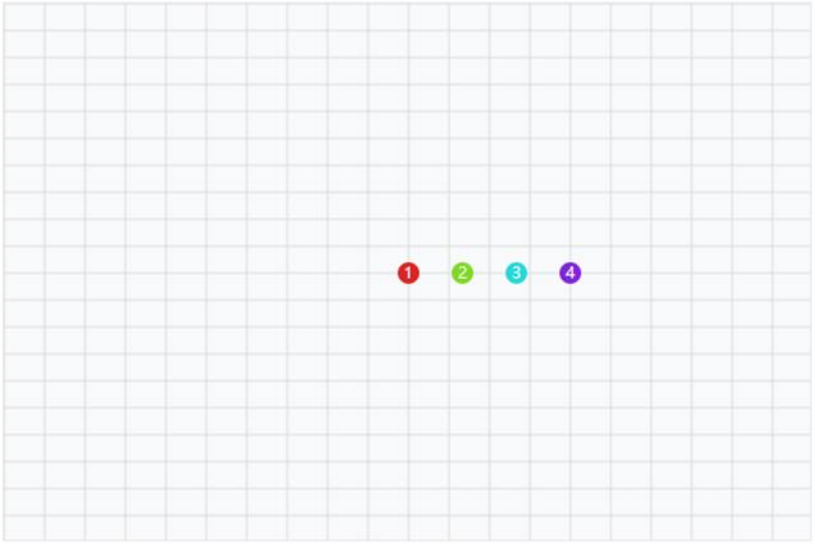
In this study, Claude was used to write a "formation flight control system" for drones and use AI to assist in operation. In the program, there are 6 ways to form a drone, and you can see the drone's movement trajectory as it moves from team A to team B.

3.1 Achievement display

進階無人機編隊控制系統

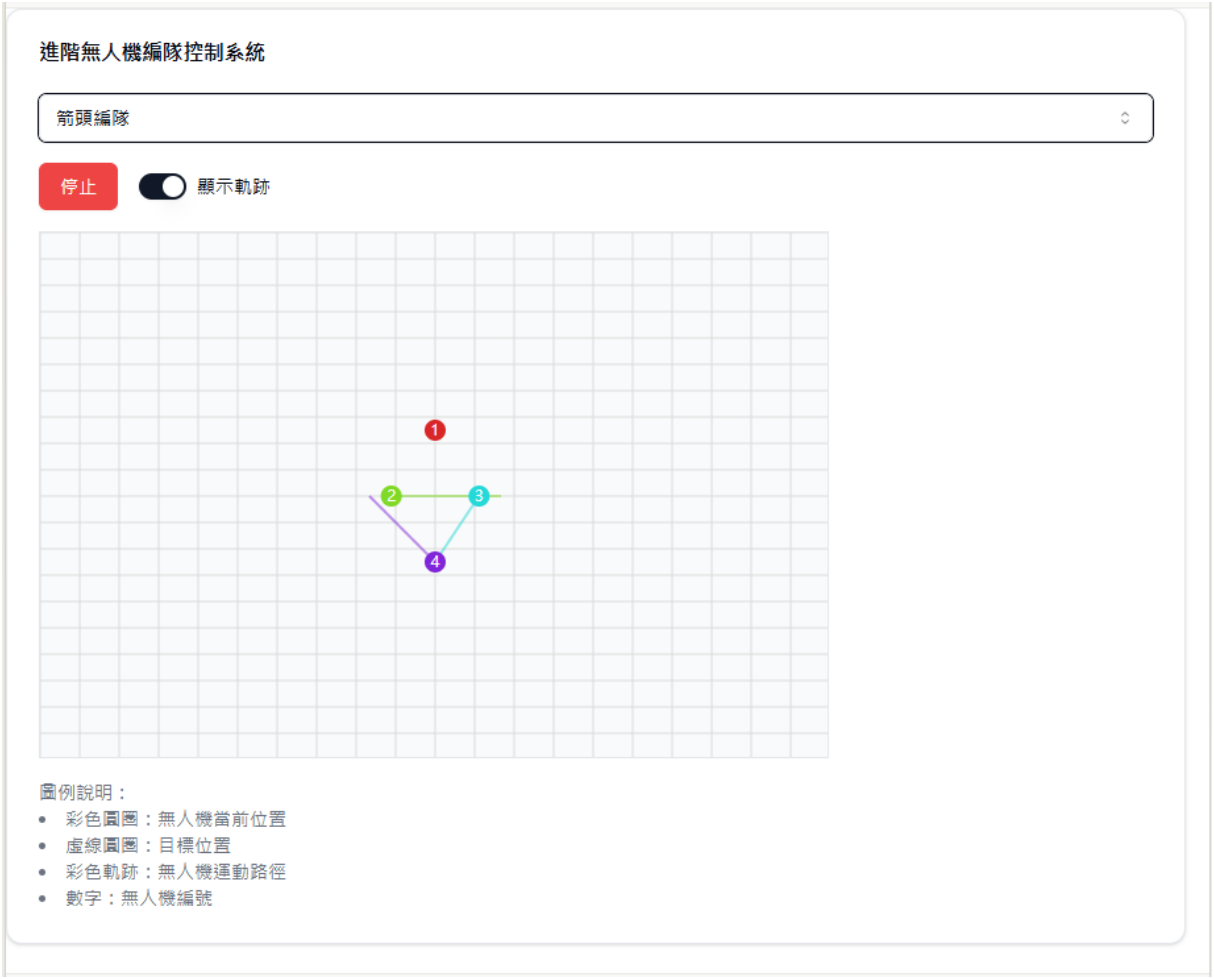
三角形編隊

開始 顯示軌跡



圖例說明：

- 彩色圓圈：無人機當前位置
- 虛線圓圈：目標位置
- 彩色軌跡：無人機運動路徑
- 數字：無人機編號



This program has the following features:

(1) Stand-alone control

- Position and speed control
- Safe distance maintenance
- Collision avoidance
- Maximum speed and acceleration limits

(2) Formation control

- Support multiple formation shapes (triangle, circle, ...)etc.)
- Real-time location updates
- Smooth formation changes
- Formation status monitoring

(3) Safety features

- Collision Avoidance Algorithm
- Speed and acceleration limits
- Keep a safe distance

(4) Trajectory display

- Each drone has a unique color movement trajectory
- Track uses a gradual transparency effect to make it easier to observe the direction of movement
- Keep the last 100 location points to form a trajectory

3.2 Questions

(1) It can only control the UAV formation, and cannot deal with abnormal conditions in time

(2) It can display the movement trajectory, but cannot show the process of drone flight

3.3 Solutions

(1) Adjust the safety distance according to actual needs (safety_distance)

(2) The maximum speed (max_velocity) and acceleration (max_acceleration) can be adjusted according to the performance of the drone

(3) The update frequency (DT) can be adjusted according to the requirements of the control system

Chapter IV: Conclusions and Recommendations

In this special feature, I hope to develop an "AI system", but there is still a big gap between the actual design and the "real AI".

The first time I did a special report, it was really a super headache. Although I am studying "drone learning", it is still not easy to make "professional achievements" related to drones, and I will choose the theme this time because of the various themes that come to mind, only "drone formation" is more suitable for play; However, through this experience, I also learned how to produce a special feature, which is a big step forward in my confidence in making a graduation feature in the future.

At present, it is a major deficiency of this program to allow drones to make formation adjustments and discharge different teams, but they cannot make their own judgments on the anomalies of the environment. In Claude, if you only use a free membership, you can only ask up to 5 questions on a topic, and you can't ask questions again after that, and

Claude is an "AI assistant tool", even if you ask the same question, it will run different results, resulting in if the program is wrong, it will be difficult to correct it, and you can only change it from scratch; In order to get Claude to run the "correct program", he can only get the desired result by asking questions about the system many times, finding out the problems in the system, and correcting them step by step.

References

【1】Claude

https://claude.ai/chat/70bbb7db-a608-44d2-af8c-dfc00fbca486

【2】Application of Artificial Intelligence in Flying Vehicles: What Are the Differences in the Development Priorities of UAVs and Civil Aircraft?

https://scitechvista.nat.gov.tw/Article/c000003/detail?ID=2c82f39a-f9fe-43c6-9966-d53e56a111d5

【3】The US MQM-178 UAV completed the flight test of the self-assembled team

https://www.ydn.com.tw/news/newsInsidePage?chapterID=1702952&type=vision

【4】Seize the golden hour! 5G drones combined with AI recognition to strengthen search and rescue in mountain disasters

https://ecos.org.tw/ad-co.php?keyValue=1qZISS7n9r3j1XHK3f61MA%3D%3D

【5】Autonomous drone inspection system

https://www.twaicoe.org/%E8%87%AA%E4%B8%BB%E7%84%A1%E4%BA%BA%E6%A9%9F%E5%B7%A1%E6%AA%A2%E7%B3%BB%E7%B5%B1

【6】Claude (Chatbots)

https://zh.wikipedia.org/zh-tw/Claude_(%E8%81%8A%E5%A4%A9%E6%A9%9F%E5%99%A8%E4%BA%BA)

【7】Welcome to Claude

https://docs.anthropic.com/zh-TW/docs/welcome

Program

```
import React, { useState, useEffect } from 'react';
import { Card, CardHeader, CardTitle, CardContent } from
'@/components/ui/card';
import { Button } from '@/components/ui/button';
import { Switch } from '@/components/ui/switch';
import { Label } from '@/components/ui/label';
import {
  Select,
  SelectContent,
```

```
  SelectItem,
  SelectTrigger,
  SelectValue,
} from "@/components/ui/select";

// Drone 類保持不變...
class Drone {
  constructor(id, position) {
    this.id = id;
    this.position = position;
    this.velocity = [0, 0, 0];
    this.targetPosition = [...position];
    this.maxVelocity = 2.0;
    this.maxAcceleration = 1.0;
    this.safetyDistance = 2.0;
    this.trajectoryPoints = [position];
  }

  updatePosition(dt, nearbyDrones) {
    const direction = this.targetPosition.map((t, i) => t -
```

```
this.position[i]);
    const distance = Math.sqrt(direction.reduce((sum, d) => sum + d * d,
0));

    if (distance > 0) {
      const normalizedDirection = direction.map(d => d / distance);
      this.position = this.position.map((p, i) =>
        p + normalizedDirection[i] * Math.min(this.maxVelocity * dt,
distance));
      this.trajectoryPoints.push([...this.position]);
      if (this.trajectoryPoints.length > 100) {
        this.trajectoryPoints.shift();
      }
    }
  }
}

const DroneFormationVisualization = () => {
  const [drones, setDrones] = useState([]);
  const [formation, setFormation] = useState('triangle');
```

```
const [isRunning, setIsRunning] = useState(false);
const [showTrajectory, setShowTrajectory] = useState(true);
const [formationSize, setFormationSize] = useState(5.0); // 編隊大小

const width = 600;
const height = 400;
const scale = 20;
const centerX = width / 2;
const centerY = height / 2;

useEffect(() => {
  const initialDrones = Array(4).fill(null).map((_, i) =>
    new Drone(i, [i * 2, 0, 10])
  );
  setDrones(initialDrones);
}, []);

// 擴展編隊形狀設置函數
const setFormationTargets = (formationType) => {
  const newDrones = [...drones];
```

```
switch (formationType) {
  case 'triangle':
    const height = formationSize * Math.sqrt(3) / 2;
    const trianglePositions = [
      [0, 0, 10],
      [formationSize/2, height, 10],
      [-formationSize/2, height, 10],
      [0, height * 1.5, 10]
    ];
    newDrones.forEach((drone, i) => {
      drone.targetPosition = trianglePositions[i];
    });
    break;

  case 'square':
    const squarePositions = [
      [-formationSize/2, -formationSize/2, 10],
      [formationSize/2, -formationSize/2, 10],
      [formationSize/2, formationSize/2, 10],
```

```
    [-formationSize/2, formationSize/2, 10]
  ];
  newDrones.forEach((drone, i) => {
    drone.targetPosition = squarePositions[i];
  });
  break;

case 'diamond':
  const diamondPositions = [
    [0, formationSize/2, 10],
    [formationSize/2, 0, 10],
    [0, -formationSize/2, 10],
    [-formationSize/2, 0, 10]
  ];
  newDrones.forEach((drone, i) => {
    drone.targetPosition = diamondPositions[i];
  });
  break;

case 'line':
```

```
    const spacing = formationSize / (newDrones.length - 1);
    newDrones.forEach((drone, i) => {
      drone.targetPosition = [
        -formationSize/2 + i * spacing,
        0,
        10
      ];
    });
    break;

  case 'circle':
    newDrones.forEach((drone, i) => {
      const angle = (2 * Math.PI * i) / newDrones.length;
      drone.targetPosition = [
        formationSize * Math.cos(angle),
        formationSize * Math.sin(angle),
        10
      ];
    });
    break;
```

```
    case 'arrow':
      const arrowPositions = [
        [0, formationSize/2, 10],  // 箭頭尖
        [-formationSize/3, 0, 10], // 左翼
        [formationSize/3, 0, 10],  // 右翼
        [0, -formationSize/2, 10]  // 尾部
      ];
      newDrones.forEach((drone, i) => {
        drone.targetPosition = arrowPositions[i];
      });
      break;
  }

  setDrones(newDrones);
};

// 重置軌跡
const resetTrajectories = () => {
  setDrones(prevDrones => prevDrones.map(drone => {
```

```
    const newDrone = new Drone(drone.id, [...drone.position]);
    newDrone.targetPosition = [...drone.targetPosition];
    newDrone.trajectoryPoints = [[...drone.position]];
    return newDrone;
  }));
};

// 更新無人機位置
useEffect(() => {
  let animationFrame;
  const updatePositions = () => {
    if (isRunning) {
      setDrones(prevDrones => {
        const newDrones = prevDrones.map(drone => {
          const newDrone = new Drone(drone.id, [...drone.position]);
          newDrone.targetPosition = [...drone.targetPosition];
          newDrone.trajectoryPoints = [...drone.trajectoryPoints];
          newDrone.updatePosition(0.1, prevDrones);
          return newDrone;
        });
```

```
        return newDrones;
      });
      animationFrame = requestAnimationFrame(updatePositions);
    }
  };

  if (isRunning) {
    animationFrame = requestAnimationFrame(updatePositions);
  }

  return () => {
    if (animationFrame) {
      cancelAnimationFrame(animationFrame);
    }
  };
}, [isRunning]);

const getTrajectoryPath = (points) => {
  if (points.length < 2) return '';
  return points.reduce((path, point, index) => {
```

```
    const command = index === 0 ? 'M' : 'L';
    const x = centerX + point[0] * scale;
    const y = centerY - point[1] * scale;
    return `${path} ${command} ${x},${y}`;
  }, '');
};

return (
  <Card className="w-full max-w-4xl">
    <CardHeader>
      <CardTitle>進階無人機編隊控制系統</CardTitle>
    </CardHeader>
    <CardContent>
      <div className="flex flex-col gap-4">
        <div className="flex flex-wrap gap-4 items-center">
          <Select
            value={formation}
            onValueChange={(value) => {
              setFormation(value);
              setFormationTargets(value);
```

```
      resetTrajectories();
    }}
  >
    <SelectTrigger className="w-[180px]">
      <SelectValue placeholder="選擇編隊形狀" />
    </SelectTrigger>
    <SelectContent>
      <SelectItem value="triangle">三角形編隊</SelectItem>
      <SelectItem value="square">方形編隊</SelectItem>
      <SelectItem value="diamond">菱形編隊</SelectItem>
      <SelectItem value="circle">圓形編隊</SelectItem>
      <SelectItem value="line">直線編隊</SelectItem>
      <SelectItem value="arrow">箭頭編隊</SelectItem>
    </SelectContent>
  </Select>

  <Button
    onClick={() => {
      setIsRunning(!isRunning);
      if (!isRunning) resetTrajectories();
```

```
      }}
      variant={isRunning ? 'destructive' : 'default'}
    >
      {isRunning ? '停止' : '開始'}
    </Button>

    <div className="flex items-center space-x-2">
      <Switch
        id="trajectory-switch"
        checked={showTrajectory}
        onCheckedChange={setShowTrajectory}
      />
      <Label htmlFor="trajectory-switch">顯示軌跡</Label>
    </div>
  </div>

  <svg
    width={width}
    height={height}
    className="border border-gray-200 bg-gray-50"
```

```
>
  {/* 網格 */}
  {Array(20).fill(null).map((_, i) => (
    <React.Fragment key={i}>
      <line
        x1={0}
        y1={i * (height/20)}
        x2={width}
        y2={i * (height/20)}
        stroke="#ddd"
        strokeWidth="1"
      />
      <line
        x1={i * (width/20)}
        y1={0}
        x2={i * (width/20)}
        y2={height}
        stroke="#ddd"
        strokeWidth="1"
      />
```

```
  </React.Fragment>
))}

{/* 無人機和軌跡 */}
{drones.map((drone) => (
  <g key={drone.id}>
    {showTrajectory && (
      <path
        d={getTrajectoryPath(drone.trajectoryPoints)}
        stroke={`hsl(${drone.id * 90}, 70%, 50%)`}
        strokeWidth="2"
        fill="none"
        opacity="0.5"
      />
    )}

    <circle
      cx={centerX + drone.targetPosition[0] * scale}
      cy={centerY - drone.targetPosition[1] * scale}
      r="4"
```

```
  fill="none"
  stroke="#666"
  strokeDasharray="2,2"
/>

<circle
  cx={centerX + drone.position[0] * scale}
  cy={centerY - drone.position[1] * scale}
  r="8"
  fill={`hsl(${drone.id * 90}, 70%, 50%)`}
/>
<text
  x={centerX + drone.position[0] * scale}
  y={centerY - drone.position[1] * scale + 4}
  textAnchor="middle"
  fill="white"
  fontSize="12"
>
  {drone.id + 1}
</text>
```

```
            </g>
          ))}
        </svg>

        <div className="text-sm text-gray-500">
          <p>圖例說明：</p>
          <ul className="list-disc list-inside">
            <li>彩色圓圈：無人機當前位置</li>
            <li>虛線圓圈：目標位置</li>
            <li>彩色軌跡：無人機運動路徑</li>
            <li>數字：無人機編號</li>
          </ul>
        </div>
      </div>
    </CardContent>
  </Card>
 );
};

export default DroneFormationVisualization;
```