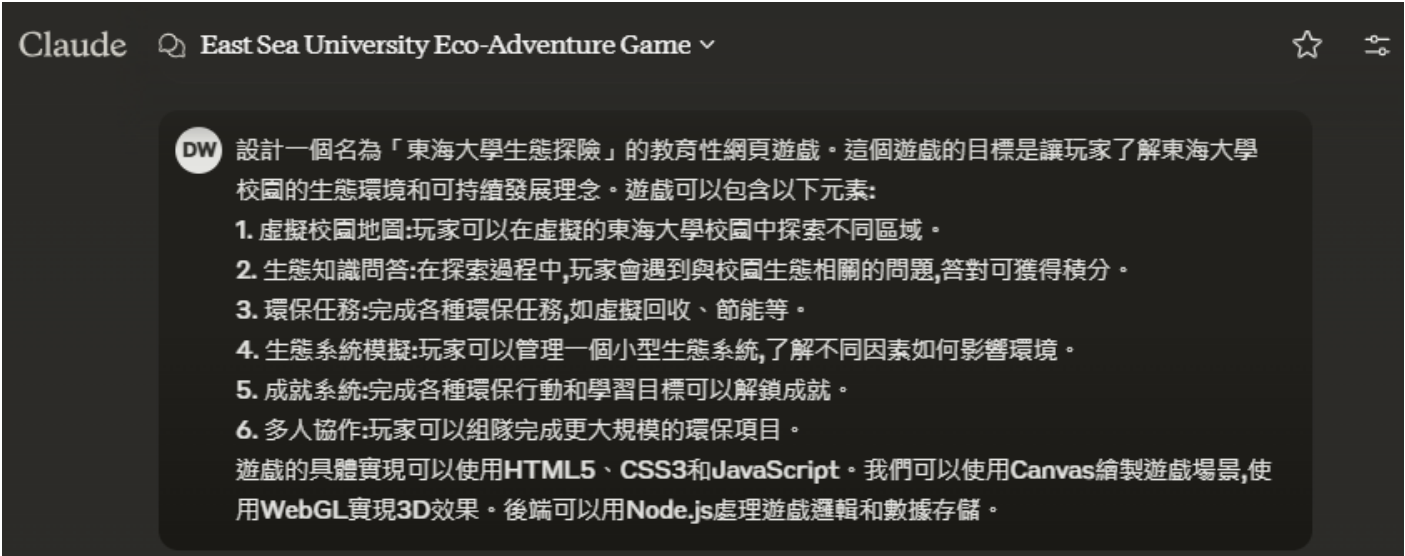


# Claude AI



## Claude AI

1. 設計一個名為「東海大學生態探險」的教育性網頁遊戲。這個遊戲的目標是讓玩家了解東海大學校園的生態環境和可持續發展理念。遊戲可以包含以下元素:

1. 虛擬校園地圖: 玩家可以在虛擬的東海大學校園中探索不同區域。

```
javascript
// 初始化
class EcoAdventure {
  constructor() {
    this.map = new VirtualMap();
    this.player = new Player();
    this.quizSystem = new QuizSystem();
    this.taskSystem = new TaskSystem();
    this.ecoSystem = new EcoSystem();
    this.achievementSystem = new AchievementSystem();
    this.multiplayerSystem = new MultiplayerSystem();
  }

  init() {
    // 初始化
  }

  update() {
    // 更新
  }

  render() {
    // 渲染
  }
}
```

## 2. 虚拟地图

javascript

```
class VirtualMap {
  constructor() {
    this.areas = [
      { name: 'A', x: 0, y: 0 },
      { name: 'B', x: 100, y: 100 },
      { name: 'C', x: 200, y: 50 },
      // ...
    ];
  }

  render() {
    // Canvas WebGL
  }

  movePlayer(x, y) {
    // ...
  }
}
```

## 3. 问答系统

javascript

```
class QuizSystem {
  constructor() {
    this.questions = [
      {
        question: '1+1=?',
        options: ['A: 1', 'B: 2', 'C: 3', 'D: 4'],
        answer: 0,
      },
      // ...
    ];
  }

  askQuestion() {
    // ...
  }

  checkAnswer(questionId, answerId) {
    // ...
  }
}
```

## 4. 任务系统

javascript

```
class TaskSystem {
  constructor() {
    this.tasks = [
      { id: 1, name: 'Task 1', completed: false },
      { id: 2, name: 'Task 2', completed: false },
      // ...
    ];
  }
```

```
}

assignTask() {
    // TODO
}

completeTask(taskId) {
    // TODO
}
}
```

## 5. 任务系统

javascript

```
class EcoSystem {
    constructor() {
        this.factors = {
            temperature: 25,
            humidity: 60,
            pollution: 10
        };
        this.flora = [];
        this.fauna = [];
    }

    update() {
        // TODO
    }

    addOrganism(type, species) {
        // TODO
    }

    removeOrganism(type, species) {
        // TODO
    }
}
```

## 6. 生态系统

javascript

```
class AchievementSystem {
    constructor() {
        this.achievements = [
            { id: 1, name: '1000 Points', description: '1000 Points', unlocked: false },
            { id: 2, name: '500 Points', description: '500 Points', unlocked: false },
            // ...
        ];
    }

    checkAchievements(player) {
        // TODO
    }
}
```

## 7. 成就系统

javascript

```
class MultiplayerSystem {
  constructor() {
    this.players = [];
    this.teams = [];
    this.projects = [];
  }

  createTeam(players) {
    // TODO
  }

  startProject(team, projectType) {
    // TODO
  }

  updateProjectProgress(projectId, progress) {
    // TODO
  }
}
```

TODO:

- 1. Implement React/Vue.js components
- 2. Integrate Three.js for 3D rendering
- 3. Implement Socket.io for real-time communication
- 4. Integrate Express.js, MongoDB for backend

Implementing a multiplayer system involves several key components and steps:

1. **Backend Setup:** Choose a framework like Node.js/Express.js and a database like MongoDB.

---

1. **Frontend Setup:** Choose a framework like React or Vue.js.

2. **Backend Setup:** Choose a framework like Node.js/Express.js and a database like MongoDB.

3. **Frontend Setup:** Choose a framework like React or Vue.js.