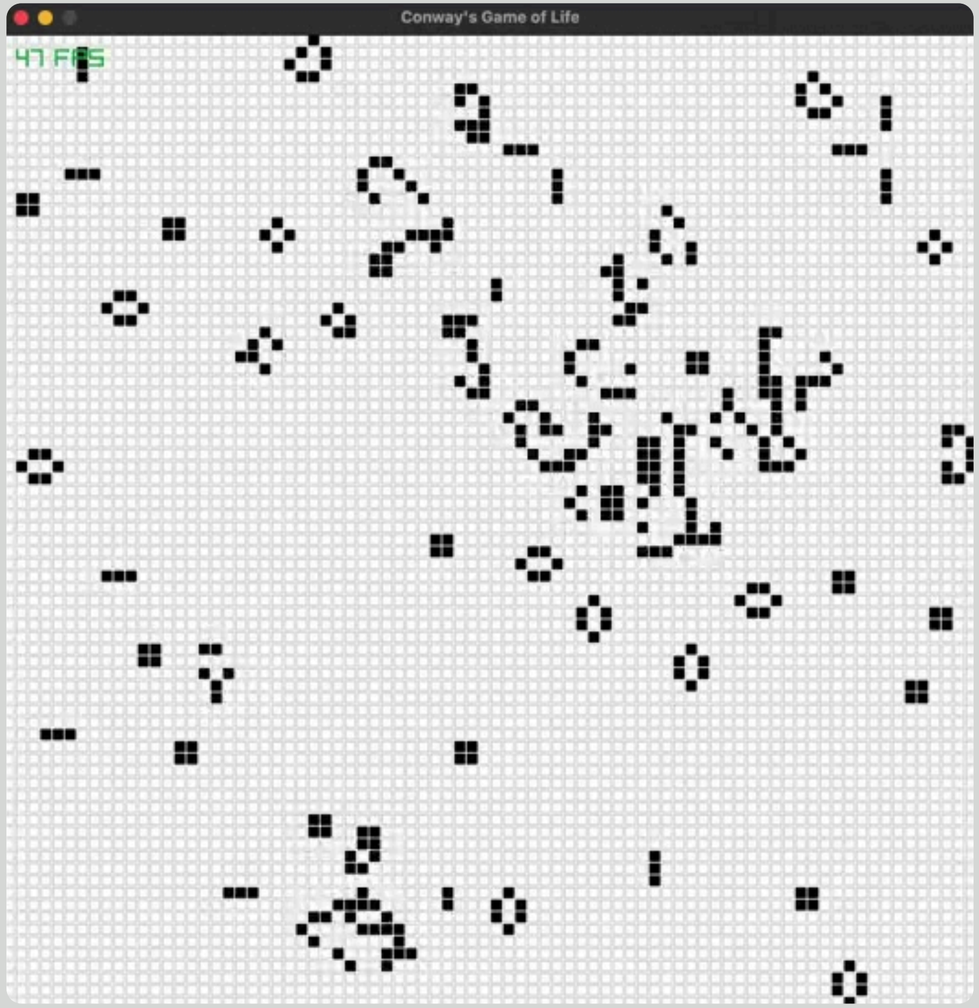
人工智慧導論

Presented by 周霖　李瑞紘 黃皓群

Computer Science 2024 | NSYSU

康威生命遊戲

康威生命遊戲



康威生命遊戲


```
public:
    GameOfLife(){//constructor…

    void addPufferTrain(){//setting the initial y of puffer train…

    void updatePufferPosition(){//calculating the next moment of the train…

    int countNeighbors(int x, int y){//calculating the neighbors in the 3*3 square…

    void update(){…




    void draw(){//drawing function…
};
```

康威生命遊戲


```
int main(){
    int fps = 30;
    InitWindow(SCREEN_WIDTH, SCREEN_HEIGHT, "Conway's Game of Life");
    SetTargetFPS(fps); //setting fps of the game
    GameOfLife game;
    while(!WindowShouldClose()){
        if(IsKeyPressed(KEY_P)){…
        if(IsKeyPressed(KEY_COMMA) && fps >= 20){…
        if(IsKeyPressed(KEY_PERIOD) && fps <= 150){…


        game.update();
        //drawing
        BeginDrawing();

        ClearBackground(WHITE);
        game.draw();

        DrawFPS(10, 10);

        EndDrawing();//end drawing
    }

    CloseWindow();
    return 0;
}
```

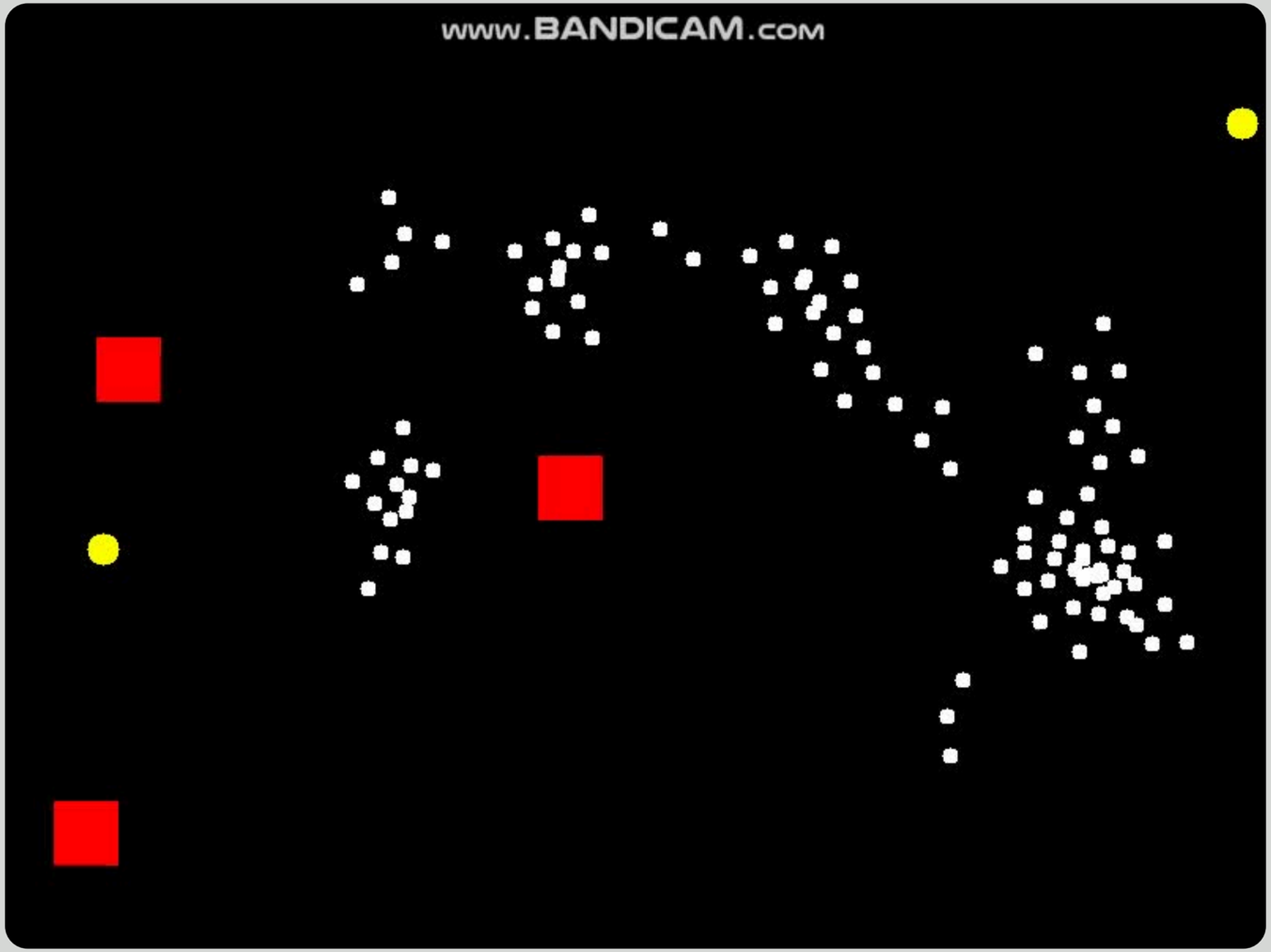
康威生命遊戲


```
class GameOfLife{
private:
    vector<vector<bool>> grid;//due to rule 1, we use bool vector to store the cell grid
    vector<vector<bool>> nextGrid;//the grid next moment
    bool hasPuffer = false;//to store if there's puffer train on the screen
    const vector<vector<bool>> pattern = {//pattern of puffer train
        {0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1},
        {1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},
        {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},
        {0,0,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    };

    int pufferY = GRID_HEIGHT - 1;//the initial y of puffer train
```

Boids鳥類群體行為模擬

Boids鳥類群體行為模擬



Boids鳥類群體行為模擬

1.分離行為 (Separation):
當鳥群靠近其他鳥或障礙物時，鳥會調整方向遠離其他個體，以避免碰撞，確保鳥群的安全性。

2.對齊行為 (Alignment):
個體根據周圍的鄰居的速度調整自身速度，以達到與周圍鳥群一致的方向和速度，保持鳥群的整齊性。

3.凝聚行為 (Cohesion):
調整自身的速度和方向，朝向鄰居鳥群的平均中心位置移動，以保持群體的凝聚性和穩定性。

4.食物尋找 (Seek Food):
當感知到食物時，鳥群會朝向最近的食物移動，以模擬尋找食物和分食行為。

5.避開障礙物 (Avoid Obstacles):
當鳥群靠近紅色標示的障礙物時，自動調整方向以避免與障礙物碰撞，增強鳥群的靈活性和安全性。

Boids鳥類群體行為模擬


```
class Bird
{
    private:
        Vector Position;
        Vector SpeedInVector;
        double MaxSpeed; //表示最大速度 // adjustable parameter
```

Boids鳥類群體行為模擬


```
Vector Separation(const vector<Bird>& neighbors,const vector<Vector>& obstacles)
{
    Vector Adjust(0, 0); //要調整的向量值
    int counter=0;
    for(const auto& n:neighbors)
    {
        if((Position-n.Position).Distance()>0 && (Position-n.Position).Distance()<25) //感知範圍 // adjustable
        {
            Vector DiffenceBetweenTwoVectors=Position-n.Position;
            DiffenceBetweenTwoVectors=DiffenceBetweenTwoVectors.MoveAtConsistentSpeed();
            Adjust=Adjust+DiffenceBetweenTwoVectors;
            counter++;
        }
    }
```

Separation() → 計算與鄰近鳥群的方向並標準化向量，再取平均向量便修正位置

Boids鳥類群體行為模擬


```
Vector Alignment(const vector<Bird>& neighbors)
{
    Vector AvgSpeed(0,0);
    int counter=0;
    for(const auto& n:neighbors)
    {
        if((Position-n.Position).Distance()>0 && (Position-n.Position).Distance()<50) //感知範圍 // adjustable
        {
            AvgSpeed=AvgSpeed+n.SpeedInVector;
            counter++;
        }
    }
    if(counter)
    {
        AvgSpeed=AvgSpeed*(1.0/counter);
        AvgSpeed=AvgSpeed.MoveAtConsistentSpeed()*MaxSpeed;
        return AvgSpeed-SpeedInVector;
    }
    return Vector(0,0);
}
```

Alignment() → 將該隻鳥的向量更新成鄰近鳥群的平均向量

Boids鳥類群體行為模擬


```
Vector Cohesion(const vector<Bird>& neighbors)
{
    Vector Center(0,0);
    int counter=0;
    for(const auto& n:neighbors)
    {
        if((Position-n.Position).Distance()>0 && (Position-n.Position).Distance()<50) //感知範圍 // adjustable parameter
        {
            Center=Center+n.Position;
            counter++;
        }
    }
    if(counter)
    {
        Center=Center*(1.0/counter);
        return (Center-Position).MoveAtConsistentSpeed()*MaxSpeed;
    }
    return Vector(0,0);
}
```

Cohesion() → 計算二維重心位置，並將向量修正為朝該方向

Boids鳥類群體行為模擬


```
Vector SeekFood(const vector<Vector>& foods)
{
    Vector DesiredDirection(0,0);
    double closestDistance=1001.0;
    bool FoundFood=false;
    for(const auto& food:foods)
    {
        if((Position-food).Distance()<closestDistance)
        {
            closestDistance=(Position-food).Distance();
            DesiredDirection=(food-Position).MoveAtConsistentSpeed();
            FoundFood=true;
        }
    }
    if(FoundFood)
        return DesiredDirection*MaxSpeed;
    return Vector(0,0);
}
```

SeekFood()- 尋找距離最近的食物，並朝該方向修正向量

Boids鳥類群體行為模擬


```
Vector Separation(const vector<Bird>& neighbors,const vector<Vector>& obstacles)
{
    Vector Adjust(0, 0); //要調整的向量值
    int counter=0;
    for(const auto& n:neighbors)…
    for(const auto& obs:obstacles)
    {
        if((Position-obs).Distance()<80) // 如果靠近障礙物，增加分離向量 // 感知範圍 // adjustable parameter
        {
            Vector DiffenceBetweenObstacle=Position - obs;
            DiffenceBetweenObstacle=DiffenceBetweenObstacle.MoveAtConsistentSpeed();
            Adjust=Adjust+DiffenceBetweenObstacle*10.0; // 增強閃避效果
            counter++;
        }
    }

    if(counter)
        Adjust=Adjust*(1.0/counter); //取平均向量
    return Adjust;
}
```

Separation()-obstacles 當鳥群靠近障礙物時，計算並閃避向量，使鳥群避開障礙物。

Boids鳥類群體行為模擬


```
void DoAll(const vector<Bird>& neighbors,const vector<Vector>& obstacles,const vector<Vector>& foods)
{
    Vector sep = Separation(neighbors,obstacles)*1.0; // adjustable parameter
    Vector align = Alignment(neighbors)*1.5; // adjustable parameter
    Vector coh = Cohesion(neighbors)*0.3; // adjustable parameter
    Vector seekFood = SeekFood(foods)*1.2; // adjustable parameter
    SpeedInVector=SpeedInVector+sep+align+coh+seekFood;
    if(SpeedInVector.Distance()>MaxSpeed)
        SpeedInVector=SpeedInVector.MoveAtConsistentSpeed()*MaxSpeed;
}
```

DoAll()- 綜合多重行為來調整最終的向量，並保持速度在最大限制內。

Boids鳥類群體行為模擬


```
void MoveToNextPostion(double elapsedTime)
{
    if(elapsedTime<20.0) // 在前20秒生成食物(以強化顯示SeekFood和Cohesion)…
    for(auto& Bird:Boids)
        Bird.DoAll(Boids,Obstacles,Foods); // 先計算應該改變的向量
    for(auto it=Foods.begin();it!=Foods.end();)
    {
        bool eaten=false;
        for (const auto& Bird:Boids)
        {
            if ((Bird.GetPosition()-*it).Distance()<25)
            {
                eaten = true;
                break;
            }
        }
        if (eaten)
            it = Foods.erase(it); //被分食
        else
            it++;
    }
    for(auto& Bird:Boids)
        Bird.MoveToNextPostion();       // 再移動到下一位置
}
```

MoveToNextPostion()- 模擬鳥群行為，生成食物、計算行為向量、移除被吃掉的食物，並更新鳥群位置。

Thanks for Watching!