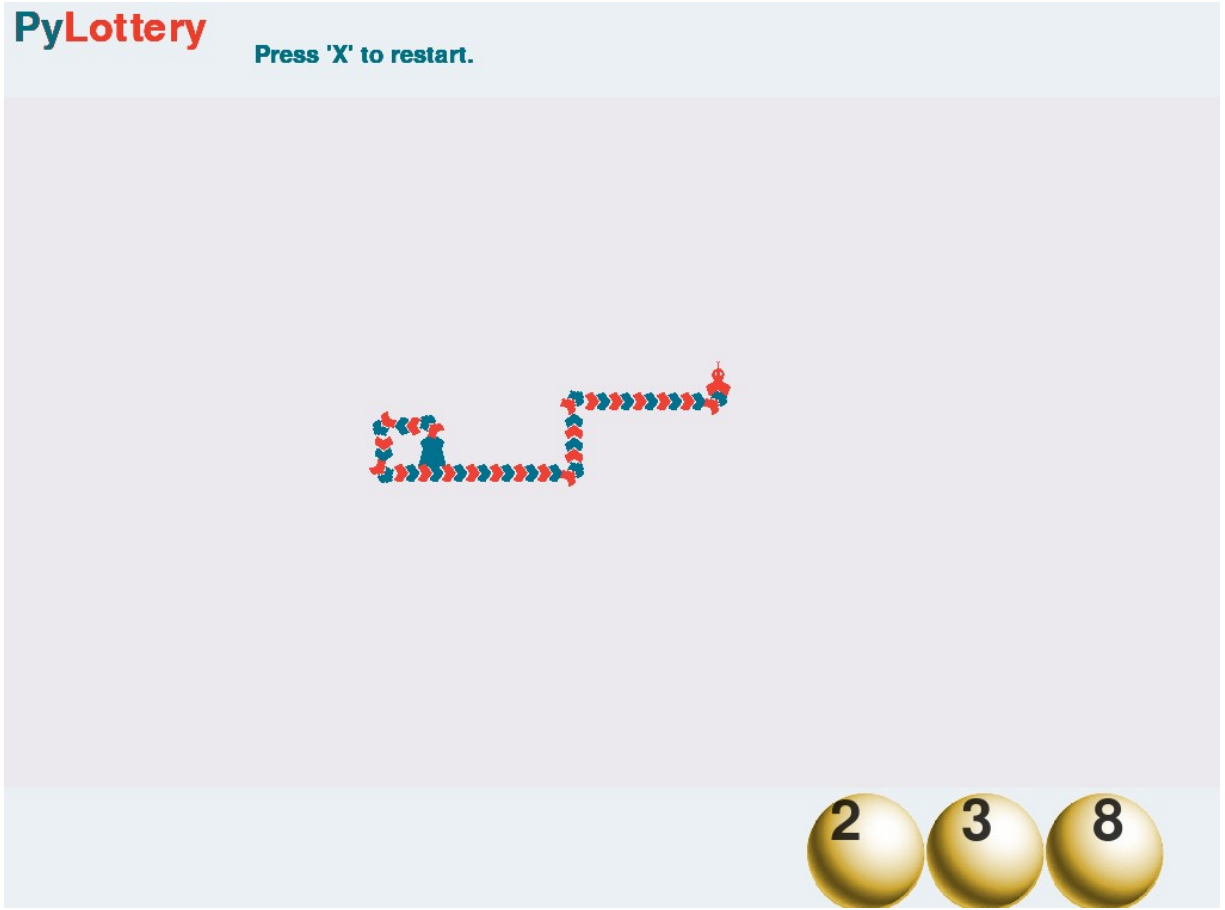
複雜系統視角下的遊戲程式設計分析

簡報大綱

- 系統概述
- 層次結構分析
- 湧現性質
- 非線性動態
- 自組織與適應性
- 隨機性與確定
- 相變與臨界現
- 系統穩健性
- 設計啟示
- 總結

PyLottery = 貪食蛇 + 彩票遊戲



簡單規則 + 複雜交互 = 湧現行為

核心特徵：

- 🐍 蛇的追蹤行為
- 🎱 隨機目標生成
- 🔄 模式動態切換
- 🎵 多感官反饋

層次結構分析

- **微觀層面（Components）**
  - **# 個體組件**
  - **snakeBody = [[x1,y1], [x2,y2], ...] # 蛇節點**
  - **ballPos = [x, y] # 目標球**
- **中觀層面（Subsystems）**
- **運動控制：**
  - **nextMove() + snakeDrawer()**
- **渲染系統：**
  - **drawSnake() + 視覺效果**
- **事件處理：**
  - **鍵盤輸入 + 狀態管理**
- **宏觀層面（Emergent Behavior）**
- **遊戲循環：整體節拍器**
- **模式切換：展示 ↔ 追獵**

湧現性質（Emergent Properties）

- **預期湧現：智能追蹤**
  - **# 簡單規則**
  - **while snakeHead != ballPos:**

    dir = 'x' if dirIdx==0 else 'y' # 交替方向

    mode = dir + ("MoveMinus" if ... else "MovePlus")
  -
  - **# 複雜行為**
  - **# → 螺旋逼近路徑**
  - **# → 看似智能的決策**
- **非預期湧現**
- **視覺韻律： 循環模式的催眠效果**
- **張力建構： 隨機性創造的不可預測感**
- **動態平衡： 自碰撞的自我修復機制**

換言之…
當觀察者「忘了」自己也是受觀察的對象之一時，他會以為發生了「非預期湧現」。

非線性動態與反饋

- **正反饋環路**
  - **# 進展 → 複雜度增加**
  - **snakeBody = snakeBody[-(10+d*10):]**
  - **遊戲進展 →**

    蛇身變長 →

    碰撞機率↑ →

    戲劇張力↑

- **負反饋環路**
- **# 自我修復機制**
- **while snakeHead in snakeBody:**

  snakeBody = snakeBody[1:] # 自動截短
- **複雜度過高 →**

  觸發簡化 →

  系統穩定↑

自組織與適應性

- **動態路徑規劃**
  - **step = randint(1, int(math.ceil(**

    abs(snakeHead[dirIdx]-ballPos[dirIdx])/50.0

    )) + 3)
  - **標度不變性：**
    - **遠距離 → 大步移動**
    - **近距離 → 精細調整**
- **方向切換機制**
- **dirIdx = 1-dirIdx # 二進制切換**
- **局部規則 → 全域螺旋模式**

隨機性與確定性的耦合

- **結構化隨機性**
  - **# 有界隨機：不是純隨機！**
  - **while abs(ballPos[0]-snakeHead[0])<80 or abs(ballPos[1]-snakeHead[1])<40:**

    ballPosX = randint(12, int(windowSize[0]/20)-12)

    ballPosY = randint(12, int((windowSize[1]-140)/20))

- **設計智慧：**

  ✅ **確保可達性**

  ✅ **維持挑戰性**

  ✅ **避免退化情況**

相變與臨界現象

- **模式轉換**
  - **展示模式 --[空格鍵]--> 追獵模式**
- **臨界行為**
- **if snakeHead in snakeBody and len(snakeBody) > 8:**

  # 臨界閾值 = 8

  # 觸發緊急處理模式

- **相變特徵：**

  🔀 **狀態急劇改變**

  📊 **閾值效應**

  ⚖️ **臨界點敏感性**

系統穩健性分析

- 穩健性機制
  - # 1. 邊界保護
  - ballPosX = randint(12, int(windowSize[0]/20)-12)
  - # 2. 自碰撞修復
  - while snakeHead in snakeBody:

    snakeBody = snakeBody[1:]
  - # 3. 資源快取
  - if imgLoaded: return # 避免重複載入

- 脆弱點
  - 資源依賴：
    - 圖片 / 音效文件缺失 → 系統崩潰
  - 硬編碼參數：缺乏動態適應能力

    (e.g., 解析度是固定的)
  - 單點故障：主循環異常 → 全系統停止

複雜性來源

- **計算複雜性**
- **狀態空間 = 位置 × 方向 × 長度 × 模式**

  = O(W×H) × 4 × L × 2

  ≈ 數千萬種可能狀態
- **認知複雜性**
  - **多時間尺度：即時反應 vs 長期策略**
  - **預測困難：隨機元素 + 非線性交互**
  - **決策樹爆炸：每步都有多種選擇路徑**

- **控制複雜性**
- **用戶輸入 → 事件處理 → 狀態更新 → 渲染 → 感知**

  ↑ ↓

  ←---------- **反饋循環** ----------←

複雜系統設計原則

- 體現的設計智慧
- 原則　　實現方式　　效果

  模塊化　　功能分離 but 適度耦合　　可維護性 ↑

  局部交互　　簡單規則 → 複雜行為　　湧現性 ↑

  多層反饋　　微觀→宏觀控制機制　　適應性 ↑

  優雅退化　　自碰撞自動修復　　穩健性 ↑

- 改進建議 (Can we make it MORE intelligent?)
- # 可引入的複雜系統特性

```
class EnhancedSnake:
    def __init__(self):
        self.learning_rate = 0.01     # 學習機制
        self.memory = []              # 路徑記憶
        self.cooperation_mode = False # 多蛇協作
        self.adaptive_params = {}     # 自適應參數
```

實際應用

- 軟體工程視角
  - 微服務架構：模塊化設計原則
  - 彈性工程：故障自恢復機制
  - 用戶體驗：隨機性與可預測性平衡
- 系統設計視角
  - 分散式系統：局部規則協調全域行為
  - 人工智慧：簡單算法產生複雜智能
  - 遊戲設計：規則簡單 but 策略豐富
- 管理學視角（人群動向 / 輿論分析）
  - 組織行為：個體交互產生團隊智慧
  - 創新管理：約束條件下的創造性湧現

總結：Simple but not Easy

- 複雜系統之核心特色在於：簡單的規則創造豐富的行為
- PyLottery 展現了：
    - 🌟 湧現之美： 簡單組件 → 複雜行為
    - 🔄 適應之智： 動態調整 + 自我修復
    - ⚖️ 平衡之道： 隨機性 + 確定性
    - 🏗️ 設計之巧： 多層次反饋機制
- 整體大於部分之和
- 簡單規則，複雜行為
- 秩序從混沌中湧現