ChatGPT

מצגת PowerPoint: הסבר על מערכת הסנסורים במשחק סוכנים - #C

שקופית 1: כותרת

שם הפרויקט: מערכת סנסורים לאיתור סוכן איראני

שפה: #C

מודולים עיקריים: SOLID, OOP, Random, Dictionary

שקופית 2: מבנה כללי של הקוד

מחלקות עיקריות:

- `Program` - נקודת הכניסה הראשית
- `Activate` - שליטה בלוגיקה של המשחק
- `Agent` - בניית הסוכן האיראני
- `GameState` - שמירה על מצב החיישנים שנפגעו
- `AgentRankBehavior` - ניהול התנהגות הסוכן לפי דרגתו
- `SensorFactory` + `Sensor` - בניית החיישנים והתנהגותם
- `Rank` - מיפוי דרגות

דיאגרמת תקשורת בין המחלקות:

```
Program → Activate → Agent + GameState + AgentRankBehavior
Agent → Rank + SensorFactory
SensorFactory → Sensor
```

שקופית 3: מחלקת Program

- **תפקיד:** נקודת התחלה של המשחק
- יוצרת מופע של `Activate`
- מבקשת קלט מהמשתמש
- מאפשרת לשחק על ידי הכנסת שמות חיישנים
- **מבנה:**
- `Main()` - מתודת ההפעלה הראשית שמריצה את המשחק בלולאת קלט-פלט מהקונסול

שקופית 4: מחלקת Activate

- **תפקיד:** ניהול כל שלבי המשחק, מהתחלה ועד ניצחון
- **אחריות:**
- יצירת סוכן חדש

- בדיקת פגיעות של השחקן
- ניהול ספירת פגיעות
- הפעלת תגובות לפי דרגת הסוכן
- **מבנה:**
- `StartGame()` - מייצרת סוכן חדש
- `FindAgent(string)` - בודקת אם הקלט תואם לחיישן, מפעילה אותו, מוסיפה ל־GameState, קוראת ל־RankBehavior
- כוללת משתנים: `agent`, `gameState`, `rankBehavior`, `count`
- **שימושים:**
- מבצעת קריאה למחלקת הדרגות כדי להבין מהן הפעולות ההתקפיות שיש לבצע לאחר פגיעות מסוימות

שקופית 5: מחלקת Agent

- **תפקיד:** בניית סוכן איראני עם דרגה וחיישנים מתאימים
- **מבנה:**
- `NewAgent()` - בוחרת דרגה אקראית מתוך `Rank` ומוסיפה חיישנים בהתאם למספר שנקבע בדרגה
- `GetAgent()` - מחזירה את המילון של החיישנים
- `RemoveSensor(string)` - מסירה חיישן מסוים מהמילון
- **חיבורים:**
- משתמש במחלקת `Rank` לקביעת דרגה
- יוצר מופע של `SensorFactory` לצורך יצירת פונקציות חיישן

שקופית 6: מחלקת GameState

- **תפקיד:** ניהול רשימת החיישנים בהם הצליח השחקן לפגוע
- **מבנה:**
- `AddScored(string)` - מוסיף לרשימה אם עוד לא קיימת
- `RemoveScored(string)` - מסיר חיישן שנפגע
- `ClearScored()` - מאפס את כל החיישנים
- `GetScored()` - מחזירה את הרשימה הנוכחית של החיישנים
- **חיבורים:**
- משמשת את `Activate` וגם את `AgentRankBehavior` כדי לדעת על אילו חיישנים בוצעה פגיעה

שקופית 7: מחלקת AgentRankBehavior

- **תפקיד:** הפעלת התקפות נגד לפי הדרגה של הסוכן
- **מבנה:**
- `FootSoldier()` - לא עושה כלום
- `SquadLeader()` - כל 3 פגיעות, מסיר חיישן אחד אקראי
- `SeniorCommander()` - כל 3 פגיעות, מסיר עד שני חיישנים
- `OrganizationLeader()` - כל 3 פגיעות, מסיר חיישן אחד; כל 10 פגיעות, מאפס את כל הרשימה
- `ExecuteRankAction(string)` - מפעיל את הפונקציה המתאימה לפי שם הדרגה
- **חיבורים:**
- מקבל הפניה ל־`GameState` כדי להשפיע על רשימת החיישנים הפגועים

שקופית 8: מחלקת Rank

- **תפקיד:** החזקה במבנה הנתונים שמכיל את שמות הדרגות ומספר החיישנים שתואמים להן
- **מבנה:**

```
Dictionary<string, int> ranks = {
 { "FootSoldier", 2 },
 { "SquadLeader", 4 },
 { "SeniorCommander", 8 },
 { "OrganizationLeader", 10 }
};
```

- `GetRank()` - מחזירה את המילון
- **חיבורים:** משמשת את מחלקת `Agent`

שקופית 9: Sensor + SensorFactory

SensorFactory:

- **תפקיד:** בניית מילון של חיישנים עם פונקציות פעולה
- **מבנה:**
- יוצר מילון `Dictionary<string, Func<string>>` מתוך מחלקת `Sensor`
- מחזיר את הרשימה לפי סוגי חיישנים
- מחלקה זו מרכזת את כלל החיישנים האפשריים ומספקת ממשק נוח וסטנדרטי לשימוש במשחק

Sensor:

- **תפקיד:** מגדיר את ההתנהגות של כל חיישן
- **מבנה:**
- `Pulse()` ו־`Motion()` - עובדים על טיימר: מוסרים את עצמם אחרי 3 הפעלות
- `Thermal()`, `Light()` - מחזירים חיישנים אקראיים קיימים
- `Magnetic()` - מציג את דרגת הסוכן
- `Audio()`, `Signal()` - פלטים פשוטים

שקופית 10: תרשים זרימה כללי של המשחק

```
Program
↳ Activate.StartGame()
  ↳ Agent.NewAgent() → Rank + SensorFactory
  ↳ SensorFactory → Sensor

↳ FindAgent(sensor)
  ↳ אם פגיעה → GameState.AddScored
              ↳ AgentRankBehavior.ExecuteRankAction
  ↳ הצגה למשתמש
```

שקופית 11: סיכום הלוגיקה

1. **המחלקה Sensor** כוללת מגוון חיישנים בעלי התנהגות שונה.
2. **SensorFactory** מאגדת את החיישנים ומייצרת מהם מילון נגיש.
3. **מחלקת Rank** מחזיקה את הדרגות ואת מספר החיישנים לפי דרגה.
4. **Agent** משתמש במחלקת Rank וב־SensorFactory כדי ליצור סוכן בעל דרגה וחיישנים.
5. **Activate** מנהלת את המשחק, בודקת את פגיעות המשתמש, ומפעילה תגובת נגד באמצעות AgentRankBehavior.
6. **AgentRankBehavior** בודקת כמה חיישנים הופעלו לפי GameState ומבצעת התקפות נגד בהתאם לדרגה.

שקופית 12: הצעות לשיפור

- שימוש ב-Interfaces במקום מחלקות קשיחות
- ניהול זמן/תורות בצורה מופרדת (Game Loop)
- עיצוב גרפי במקום Console
- הפרדה בין לוגיקה להצגה לצורך בדיקות

שקופית 13: תודה!

שאלות?

יוצר: ישראל ויצמן