

Lehem Havita - The Game

By Itay Vazana & Assaf Milner



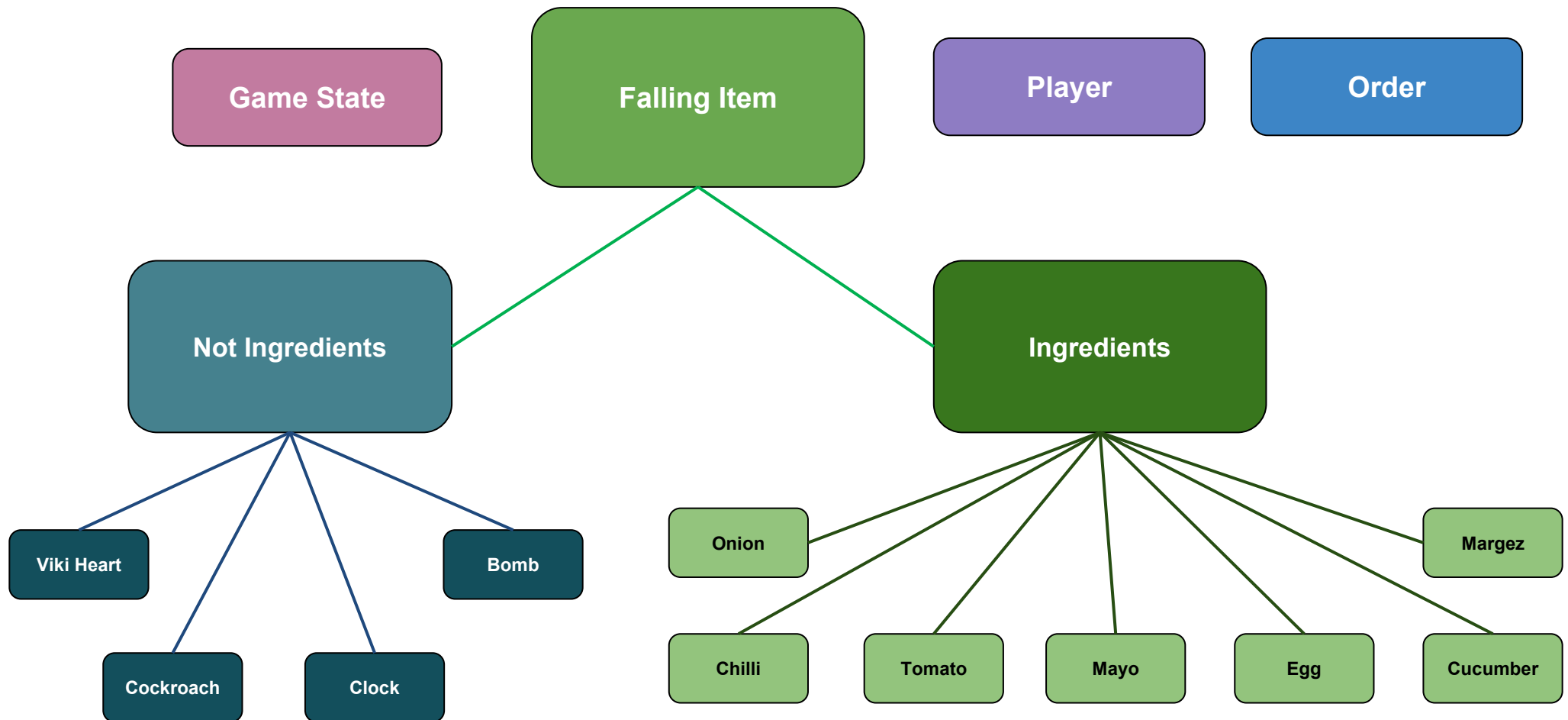
הפרויקט נכתב בשפת תכנות C# בסביבת הפיתוח Visual Studio 2019
תוך שימת דגש על..

- עקרונות של תכנות מונחה עצמים
 - לוגיקה ומשחקיות
 - חווית המשתמש
 - גרפיקה

הפרויקט הוא מחווה לטרנד ידוע שהופיע בתחילת 2023
הסובב סביב מסעדת "לחם חביטה" הממוקמת בנתניה.

"לחם חביטה – המשחק" הוא משחק מחשב לפלטפורמת PC עם מערכת הפעלה Windows
ותמיכה ב-.NET Framework. החל מגרסה 4.7.2

UML .1 – מבנה היררכי של הפרויקט



Game State

```
public static Dictionary < int, string > imagePath1 // (ingredients)
public static Dictionary < int, string > imagePath2 // (not ingredients)
public static Dictionary < int, string > Ing_name // (ingredients)
public static Dictionary < int, string > Not_Ing_names // (not ingredients)
public Falling_Item [ ] All_Ings
public Falling_Item [ ] All_Not_Ings
private List < Falling_Item > items_to_display
public Player game_player
private Order game_order
private int Current_count_on_screen
private PictureBox Player_Cursor
private Game GF
private Random rand
public System.Windows.Forms.Timer timer
public static bool Game_is_Active
```

Falling Item

```
public string Name
public PictureBox Picture
public int XPosition
public double Falling_speed
protected int Id
protected string ImagePath
private Random random
```

Ingredients : Falling Item

All Falling Item Properties and:

```
public int Cost
public virtual string Get_Name()
public virtual int Get_Cost()
```

Not Ingredients : Falling Item

All Falling Item Properties and:

```
public int Effect_On_Lives
public int Effect_On_Score
public int Effect_On_Time
public string Realeted_Sentence
public virtual void Effect_Activation(Player p)
```

Order

```
public List <string> Ing_list
public List <string> Current_Order
public int Order_count
private Game Game_Form
private Random rand
```

Player

```
public string Name
public int Score
public int Lives
```

**Bomb / Clock / Cockroach /
Viki_Heart : Not_Ingredients**

All Not Ingredients Properties and:

```
public virtual void Effect_Activation(Player p)
```

**Chilli / Egg / Cucumber / Tomato /
Margez / Onion / Mayo : Ingredients**

All Ingredients Properties and:

```
public override string Get_Name()
public override int Get_Cost()
```

Order

המחלקה אחראית על ייצוג של
ההזמנות לאורך המשחק

- טעינה מחדש של הזמנות בתחילת משחק ובכל פעם שהשחקן מסיים להכין הזמנה
- המחלקה מנהלת תקשורת שוטפת עם חלון המשחק הפעיל כדי לבצע שינויים נדרשים בתצוגת רכיבי ההזמנה בזמן אמת

Game State

המחלקה אחראית על ניהול המשחק ועל הלוגיקה
שמאחורי הפעולות השונות המתבצעות
לאורך המשחק

- ניהול זרימת המשחק מקצה לקצה
- תנועת אובייקטים
- תנועת השחקן
- ניהול אינטראקציה בין אלמנטים
- טעינה מחדש של אובייקטים וכו'...

Falling Item

המחלקה אחראית לייצוג אובייקטים "נופלים" שנעים
בציר ה-Y כלפי מטה, אותם השחקן צריך לתפוס, או
לחילופין – להימנע מפגיעה בהם

- המחלקה מורשה לשתי מחלקות:
Ingredients, Not Ingredients
- לכל אחת מהיורשים של FA יש תכונות נוספות
או מימושים של פעולות הקשורות למה שהן
מייצגות.

Player

המחלקה אחראית על ייצוג של השחקן
לאורך המשחק

מטרת העל של השחקן-

להשלים כמה שיותר הזמנות בפרק הזמן הנתון!

- אחראית לניהול האלמנטים השייכים לשחקן –
ניקוד וחיים
- המחלקה מנהלת תקשורת שוטפת עם חלון
המשחק הפעיל כדי להציג מצב עדכני של השחקן

Ingredients

המחלקה אחראית על ייצוג של FA מסוג חומרי גלם

חומרי גלם הם האובייקטים שיכולים להופיע בהזמנה
ואיסוף שלהם (**בהתאם להזמנה**) מזכה את השחקן בניקוד

מאפיין משותף שחולקים כל חומרי הגלם הוא המחיר
(השווי שלהם מבחינת ניקוד במשחק)

ועל אף שמאפיין זה משותף -
כל אחד מחומרי הגלם מגדיר
באופן ספציפי את המחיר שלו

Not_Ingredients

המחלקה אחראית על ייצוג של FA מסוג שאינו חומר גלם

כל אובייקט FA שאינו חומר גלם הוא אובייקט שאינו יכול להופיע
בהזמנה, אך עדיין הוא יכול להיכלל בגל האובייקטים הנופלים בכל
רגע נתון.

מאפיין משותף שחולקים אובייקטים מסוג זה הוא שלכולם יש
השפעה כזו או אחרת על המשחק הנוכחי בהיבט של – ניקוד, זמן,
חיים

כל אובייקט מסוג שאינו חומר גלם מממש
באופן עצמאי ויחודי לו את ההשפעה שלו על המשחק
לעיתים לטובה ולעיתים לרעה

2. חלונות קיימים במשחק



התפריט הראשי של המשחק



Rank	Player	Difficulty	Score	Sandwiches	Date Time
1	Itay	Hard	110	4	01/09/2023 12:26:07

חלונת היסטוריית ניקוד
(טבלת שיאים) עם חיבור ל-DB

Game Over !

Player - weg

Score - 9 Diff - Hard

Sandwiches - 4

חלונת המציגה בסיום
המשחק את הניקוד

Restart the Score board
Enter Administrator password:

חלונת איפוס טבלת שיאים
נדרשת סיסמא

Insert Your Name



Difficulty Level

(Default - Easy)

Easy

Mid




Hard

3 minutes - Low Speed

חלונת רישום שחקן
ובחירת רמת קושי



02:44



SCORE : 0

Tomato

Fried Egg

Chuma Chilli

Onion



Mayo

0 x



מה קשור!?



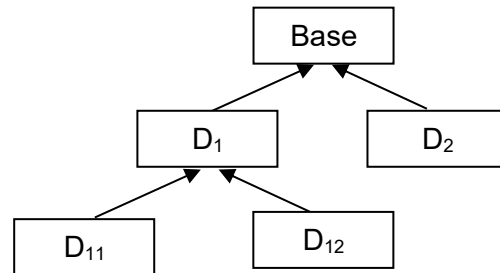


חלונית המשחק

מצד ימין
לוח המשחק עצמו

מצד שמאל
תצוגת פרטי המשחק וההזמנה הנוכחית

3. דרישות הסף בהגשה אל מול המימושים הקיימים בפרויקט



1. הפרויקט יכלול מערכת Classes מורכבת הירארכית: לא פחות מ-
הערה: המחלקה Base איננה מתוך C# או חבילה נתונה אחרת.

2. ניהול ה-Classes ע"י Polymorphism.

3. ארגון העצמים ע"י Collection object

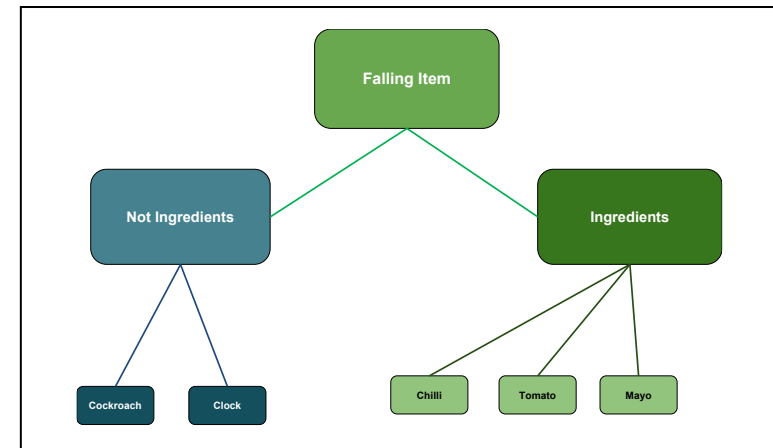
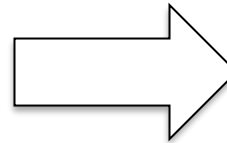
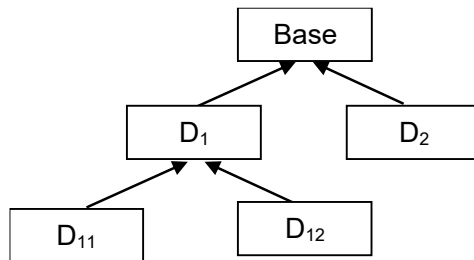
4. יכולת להוספת אובייקטים, שינוי אובייקטים ומחיקת אובייקטים.

5. תכנות מונחה אירועים (event-driven), על בסיס הטכניקות הנלמדות

6. ייצוג גרפי של אובייקטים

7. שמירת המצב הנוכחי של מערכת האובייקטים + אפשרות לשחזור ע"י מנגנון של Serialization.

1. הפרויקט יכלול מערכת Classes מורכבת הירארכית: לא פחות מ-
הערה: המחלקה Base איננה מתוך C# או חבילה נתונה אחרת



בעת יצירת מופע חדש של מחלקת ה-GameState, מופעלות הפונקציות
(Load_Paths_And_Names) ו-(Load_Arrays) שמבצעות טעינה של מילונים לנתיבים של תמונות
ולשמות ובהמשך נוצרים אובייקטים לתוך מערך מסוג Falling Item באופן הבא:

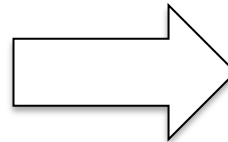
```

Falling_Item i1 = new Chilli(4);
Falling_Item i2 = new Cucumber(5);
Falling_Item i3 = new Egg(2);
....
All_Ings[0] = i1;
All_Ings[1] = i2;
All_Ings[2] = i3;
...
  
```

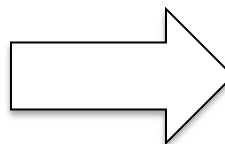
כלומר, המחלקה מממשת את עקרון הרב צורתיות בכך שהיא יוצרת לתוך מערך של אובייקטים
מסוג Falling Items אובייקטים מסוגים של מחלקת צאצא (Chilli, Cucumber..) ובהמשך
המשחק היא מטפלת בהם כ-falling item או כמחלקה יורשת אחרת (ingredients and not) בהתאמה.

בנוסף, ניתן לראות שכאשר מופעלת פונקציה המתייחסת לאובייקט מסוג falling Item
כאובייקט ממחלקה יורשת שלו ingredients / not ingredients – מופעלות הפונקציות
המוגדרות עבור מחלקות יורשות אלו, אך לא מהמחלקות יורשות עצמן אלא מהצאצאים שלהם
שכל אחד מהם מממש את הפונקציות בצורה מסוימת ועם פרמטרים רלוונטיים עבורו.

2. ניהול Classes ע"י Polymorphism



3. ארגון העצמים ע"י Collection object



בעת המשחק , מופעלת בכל תחילת "סיבוב נפילה" פונקציה מחלקת ה-GameState ששמה
() Realod_item_to_screen שמטרתה לטעון מחדש Collection object מסוג List המאפשר
:
אחסון של אובייקטים מסוג falling item , מחיקה מהרשימה של אובייקט , איתור אובייקט לפי
אינדקס , אתחול הרשימה (ריקון) וכו' ..

List הינו Generic Collection object בעל אופי דינמי , המאפשר את כל הפעולות שצוינו:

```
List<Falling_Item> items_to_display = new List<Falling_Item>();  
(יצירה של list שנועדה לאחסן אובייקטים מסוג FA)
```

```
Items_was_caught(items_to_display[i]);  
(גישה לאובייקט בתוך הרשימה דרך אינדקס i )
```

```
items_to_display.Clear();  
(ריקון הרשימה מאובייקטים)
```

```
items_to_display.Remove(items_to_display[i]);  
(הסרה של אובייקט מתוך הרשימה (גישה על ידי אינדקס) מתוך הרשימה)
```

הוספת אובייקטים –

בכל פעם שמשחק מתחיל (ע"י לחיצה של השחקן על כפתור ההתחלה ומילוי פרטים כמו שם
ורמת משחק) – נוצר אובייקט חדש של שחקן שמלווה אותו לאורך כל המשחק עד שחלון
המשחק נסגר.

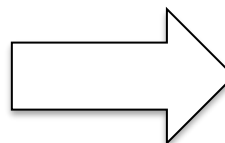
שינוי אובייקטים -

בכל "גל" של אובייקטים חדשים שנופלים – המערכת משנה את ערכי ה-X ההתחלתיים שלהם
ביחס ללוח המשחק כדי למקם אותם באופן רנדומלי.
כמו כן , בכל איסוף של אובייקט מסוג שאינו חומר גלם , מתבצע שינוי ישירות על אובייקט
השחקן עצמו (ניקוד משתנה , מד החיים משתנה וכו')

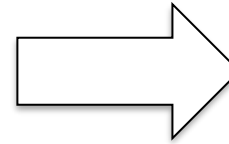
מחיקת אובייקטים –

בעת סגירת חלונות המשחק (אם באופן יזום ע"י השחקן דרך ה-X או כאשר המשחק נגמר)
מופעלת פונקציה () Kill_Game_Proccess שתפקידה לרוקן את כל המילונים והרשימות
ולמחוק את כל האובייקטים שקיימים בזמן הנוכחי.

4. יכולת להוספת אובייקטים, שינוי אובייקטים ומחיקת אובייקטים



5. תכנות מונחה אירועים (event-driven), על בסיס הטכניקות הנלמדות



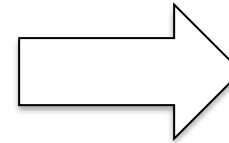
בפרויקט ישנם מגוון פונקציות שמטפלות באירועים המתרחשים לאורך הריצה של היישום :

- פונקציות לטיפול באירועי לחיצה על תמונות/כפתורים
- פונקציות לטיפול בעליות שעון (בזמן משחק רצים 2 שעונים במקביל – אחד אחראי לתזמון של תזוזת אובייקטים והשני לניהול פרק הזמן של המשחק)
- פונקציות לטיפול בקלט חומרתי (תזוזות העכבר)
- פונקציות לטיפול בהתנהגות של רכיבי WinForms (סגירה , פתיחה , טעינה..)

להלן כמה דוגמאות –

```
private void Game_MouseMove(object sender, MouseEventArgs e);  
private void Timer_Tick(object sender, EventArgs e);  
private void Pause_play_btn_Click(object sender, EventArgs e);  
private void Game_FormClosing(object sender, FormClosingEventArgs e)
```

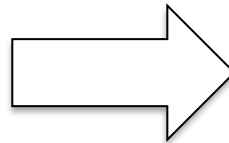
7. ייצוג גרפי של אובייקטים



לכל האובייקטים במשחק קיימת באופן כזה או אחר הצגה גרפית המיוחסת אליהם :

- לכל חומרי הגלם (Ingredients) ולכל ה-Not Ingredients (שהם סוגים של Falling Item) – קיים ייצוג גרפי ע"י תמונה שנעה על המסך.
- לשחקן – יש ייצוג גרפי ע"י תצוגת הניקוד הנוכחי שלו וכמות החיים שנותרו.
- להזמנה – ייצוג גרפי ע"י רשימה בפאנל השמאלי

6. שמירת מצב נוכחי של אובייקטים + אפשרות לשחזור ע"י Serialization



לאחר התייעצות עם המתרגל המלווה של הפרויקט וקבלת אישורו שהדבר אכן מהווה מימוש של העקרון הנדרש – החלטנו לממש את עיקרון הסריאליזציה ע"י DB מקומי.

הסריאליזציה בפרויקט שלנו ממומשת ע"י אחסון של נתוני השחקן בסוף המשחק והצגה בטבלת הניקוד שמתעדכנת מה-DB בכל פתיחה מחדש.

הקמנו בתוך תיקיית ה-Debug של הפרויקט DB file מסוג mdf שהתחברות אליו והתקשורת איתו מתבצעת ע"י הרצה של שאילתות SQL (במקרה שלנו – אחסון , עדכון , טעינה ל-DataSet).

הגורם האחראי בפרויקט לניהול התקשורת אל מול השרת בכל רגע נתון הוא חלון ה- ScoreHistoryForm שמכיל את מחרוזת ההתחברות ואת הפונקציות שמתקשרות בשאילתות SQL עם ה-DB