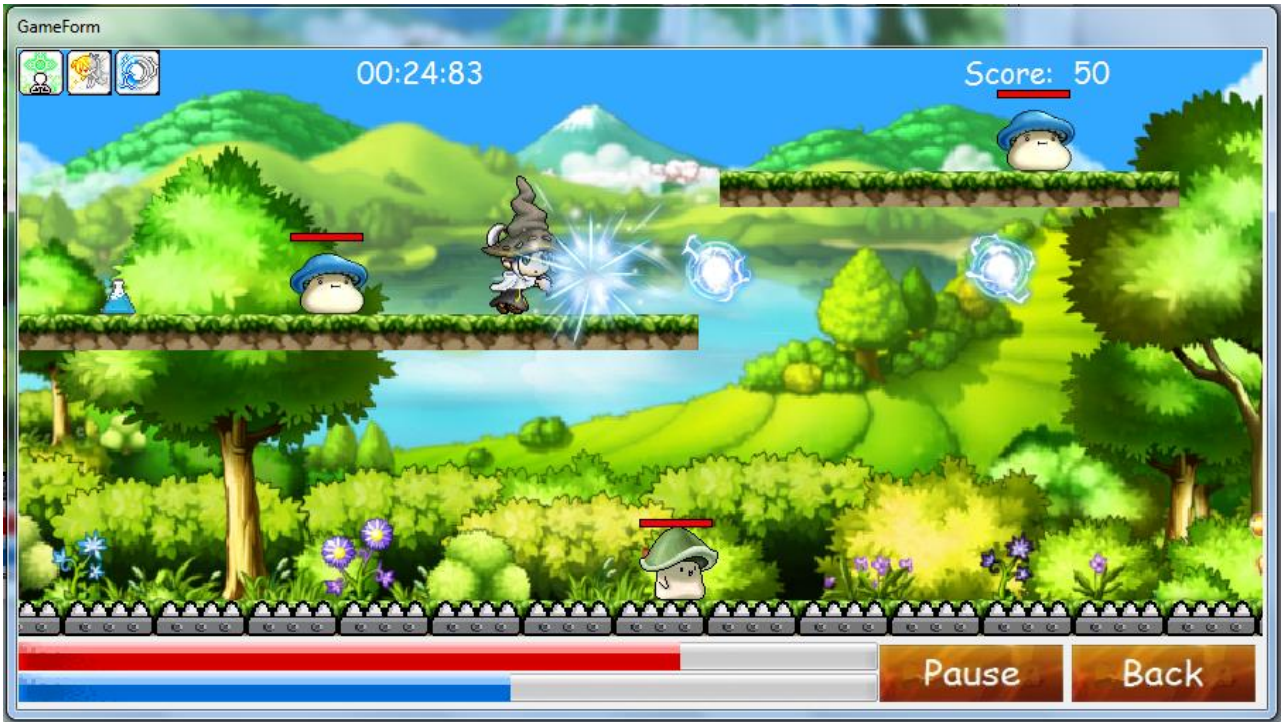
פרויקט גמר מדעי המחשב

תכנות מונחה עצמים

Object Oriented Programming

The Final Legend!



שם המגיש: ג'ונס

רמת לימוד: 5 יח"ל שפת #C

שם המורה: אולג

שם הפרוייקט: משחק אקשן בשם "The Final Legend".

תוכן עניינים

מבוא לפרויקט

המשחק שלי הוא משחק RPG (Role Playing Game), כלומר המשתמש יכול לבחור דמות שבה הוא רוצה לשחק מתוך אוסף דמויות. ב-Final Legend ישנם שלוש דמויות: Archer, Mage, Warrior. לכל דמות ישנם מאפיינים שונים: נקודות חיים, נקודות מאנה, 3 מתקפות שונות,ערך הגנה וערך התקפה. לכל התקפה יש זמן טעינה אחר שעל השחקן לחכות כדי להשתמש בה שוב. המטרה במשחק היא לעבור את השלבים שיצרתי. בשלבים ישנם מכשולים שונים: ריצפה דוקרת – אם דורכים על ריצפה זו מאבדים נקודות חיים, אויבים – אם השחקן בא במגע עם האויב הוא מאבד נקודות חיים. כאשר השחקן מאבד נקודות חיים הוא מקבל חסינות מפני פגיעה למשך של כשנייה אחת. כדי לעבור את האויבים ניתן להרוג אותם בעזרתם המתקפות של הדמות. בשימוש במתקפה נקודות מאנה יורדות. ניתן להחזיר לדמות את הנקודות חיים באסיפת אוכל או משקה מהרצפה: משקה אדום- מחזיר נקודות חיים; משקה כחול- מחזיר נקודות מאנה; המבורגר-מחזיר נקודות מאנה וחיים.

בכל רמה רמת הקושי עולה והאויבים נעשים חזקים יותר. לכל רמה ניקוד משל עצמו התלוי במשחק האויבים המתים ובזמן סיום כל רמה. בסוף המשחק השחקנים מדורגים לפי הניקוד הכללי של כל הרמות. הניקוד הכללי מחושב על ידי נוסחא.

המשחק נגמר כאשר אחד התנאים הבאים קוראים: 1. השחקן איבד את כל נקודות החיים שלו; 2. השחקן עבר את הרמה הרביעית. כאשר המשחק נגמר השחקן מועבר לטופס בו יירשום את שמו ושמו ישמר אם הניקוד שלו בין החמישה הכי טובים.

לתוכנית יש ממשק ידידותי ומספקת מענה לכל שאלותיו של השחקן.

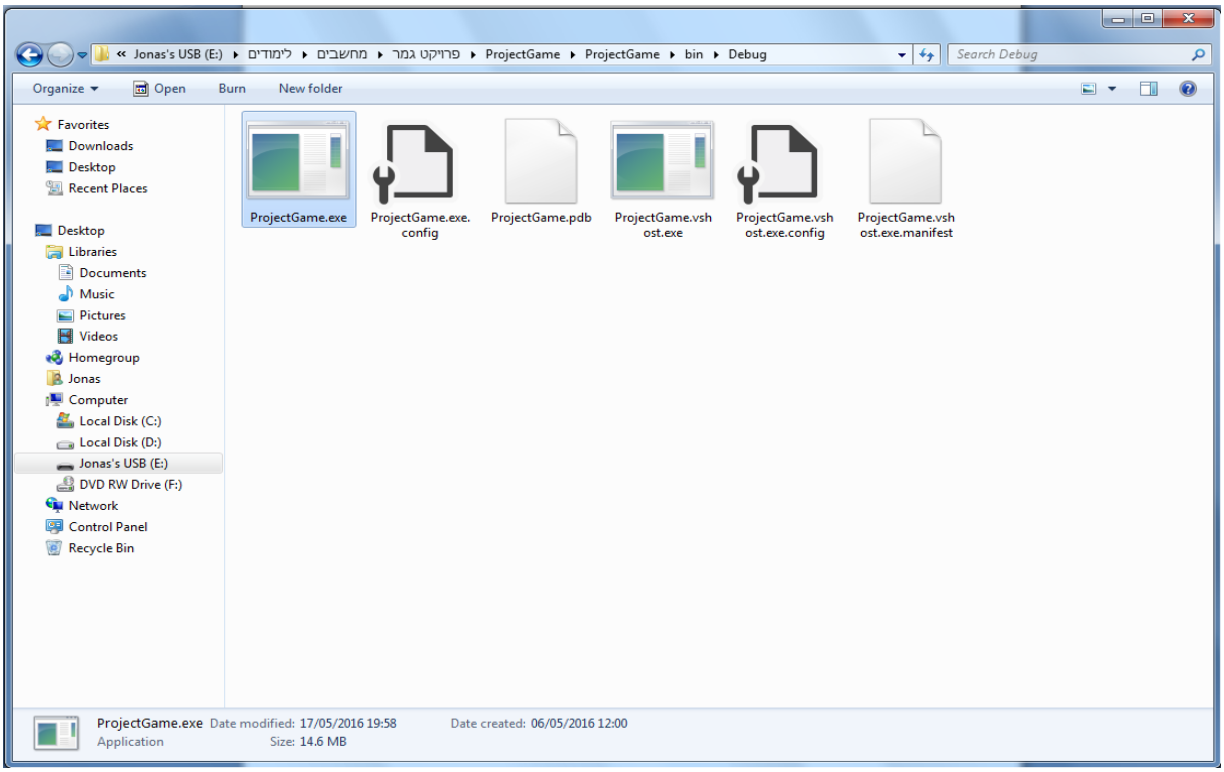
מדריך למשתמש

הפעלת המשחק

בכדי לפתוח את את המשחק יש לגשת לתיקיה debug בה נמצא קובץ המשחק שנמצאת ב:

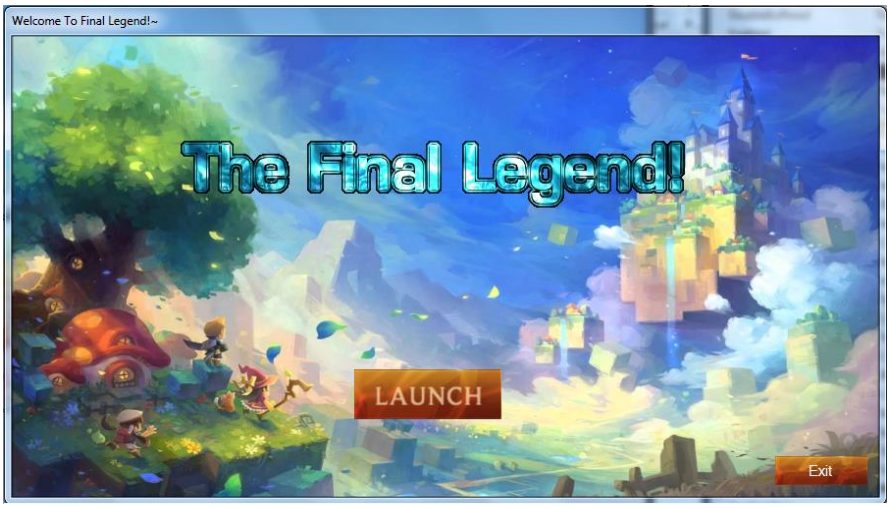
C:\ProjectGame\ProjectGame\bin\Debug\ProjectGame.Exe

כך:

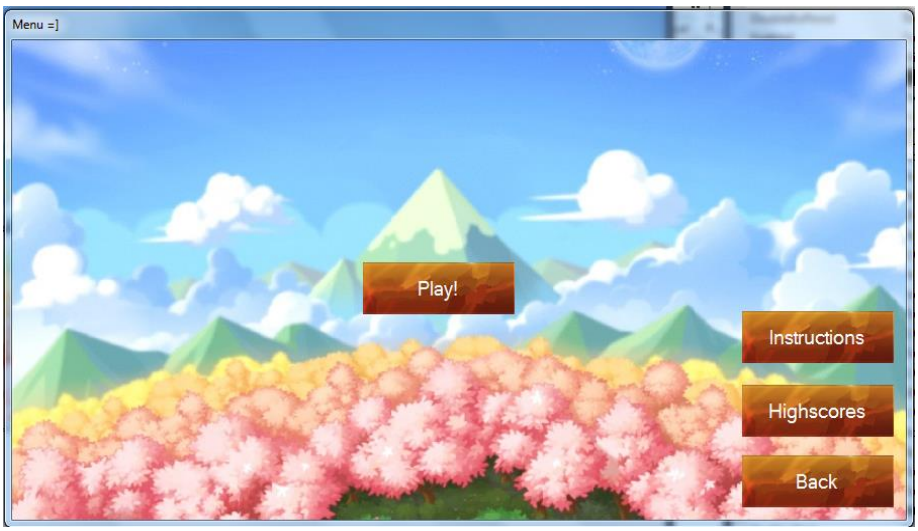


פירוט התוכנית

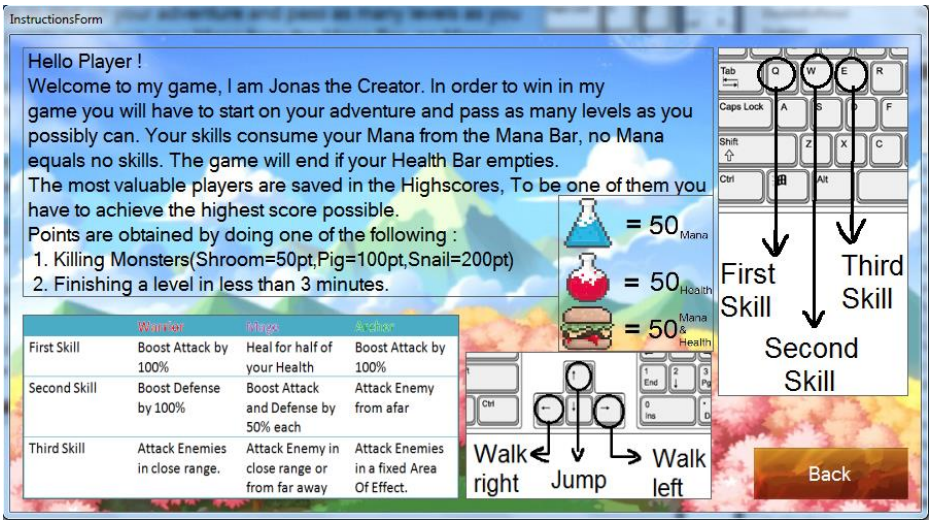
המסך שנראה בהפעלת הקובץ:



לחיצה על כפתור "Launch" תביא אותנו למסך התפריט הראשי:



לחיצה על כפתור "Instructions" תביא אותנו למסך ההסברים על המשחק:



	Wander	Kings	Archer
First Skill	Boost Attack by 100%	Heal for half of your Health	Boost Attack by 100%
Second Skill	Boost Defense by 100%	Boost Attack and Defense by 50% each	Attack Enemy from afar
Third Skill	Attack Enemies in close range.	Attack Enemy in close range or from far away	Attack Enemies in a fixed Area Of Effect.

לחיצה על הכפתור "Highscores" מהתפריט הראשי תביא אותנו למסך ה-Highscores:

לחיצה על הכפתור "Play" מהתפריט הראשי תביא אותנו למסך בחירת דמות:

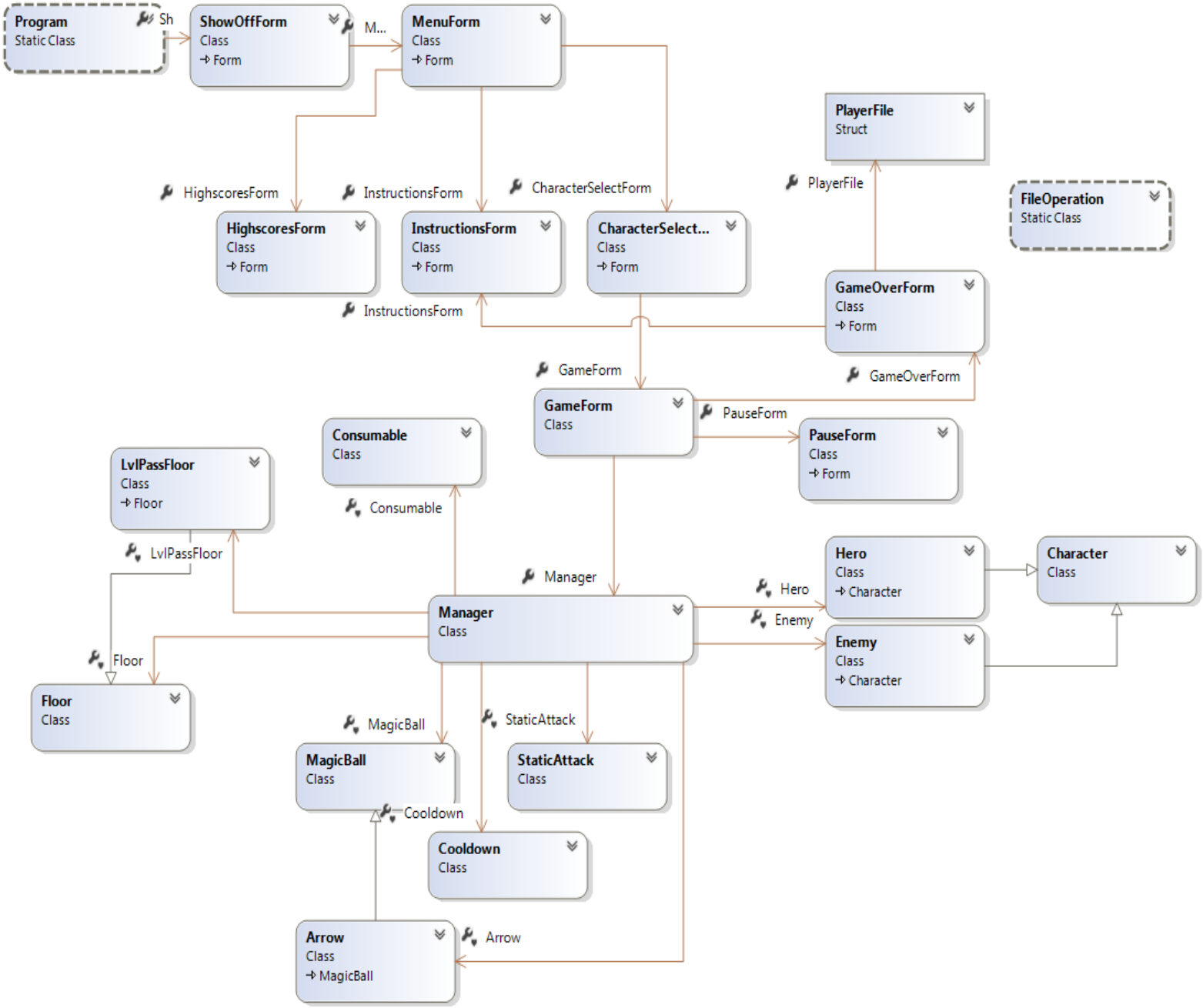
בחירת דמות ולחיצה על הכפתור "Start" תביא אותנו למסך המשחק:

בסוף המשחק אנו נגיע למסך סיום המשחק ונידרש להכניס שם תקין:

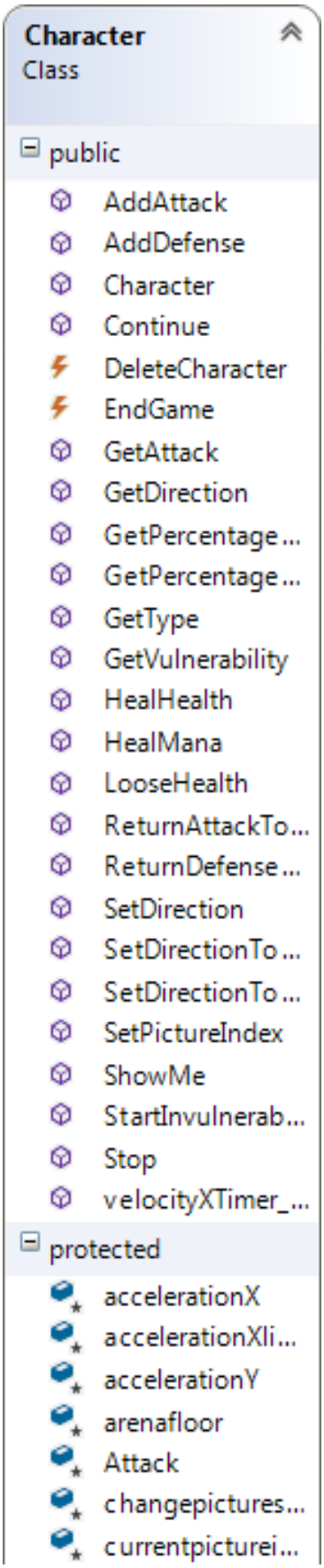
בלחיצה על כפתור "Continue" נחזור בחזרה למסך ה-Highscores.

מדריך למתכנת

דיאגרמת מבנה הפרויקט



מחלקות התוכנית

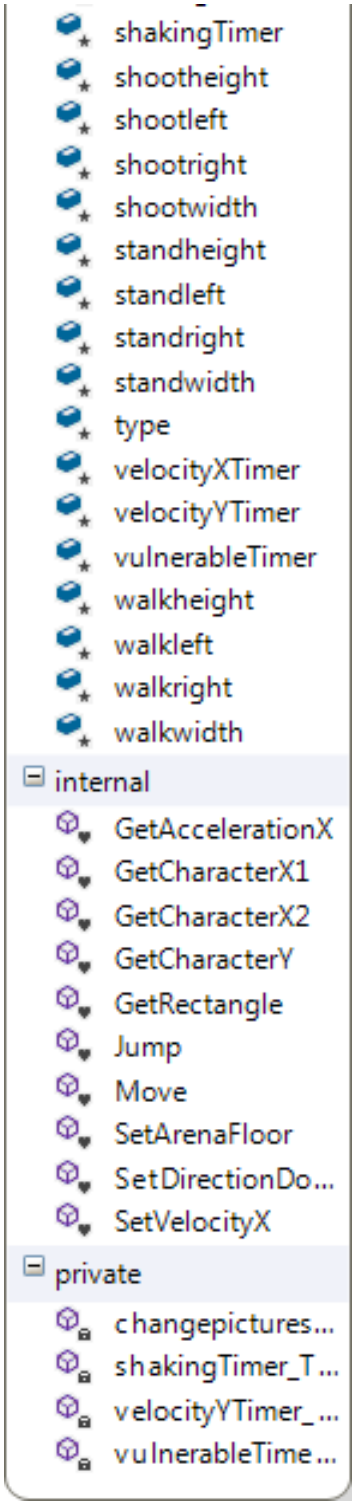


Character

```
/// <summary>
///         סוג גיבור
/// </summary>
protected CharacterType type;
/// <summary>
///         כיוון הדמות
/// </summary>
protected DirectionType direction;
/// <summary>
///         משתנה של חסינות הדמות כשהיא מקבלת מכה
/// </summary>
protected bool isVulnerable;
/// <summary>
///         מונה לכל כמה פעמים הדמות תהיה בלתי נראית
/// </summary>
protected double invulnerablecount;
/// <summary>
///         מגביל את מספר הפעמים שניתן לקפוץ
/// </summary>
protected int jumplimit;
/// <summary>
///         האם הדמות מתה
/// </summary>
protected bool isDEAD;
/// <summary>
///         תאוצה בציר האיקס
/// </summary>
protected int accelerationXlimit;
/// <summary>
///         הריצפה של הדמות,בערך הזה הדמות מפסיקה ליפול
/// </summary>
protected int arenafloor;
/// <summary>
///         גודלי התמונות בכל מצבי הדמות
/// </summary>
protected int standwidth, standheight, walkwidth,
walkheight, jumpwidth, jumpheight, downwidth, downheight,
shootwidth, shootheight, ishurtwidth,ishurtheight,
isdeadwidth, isdeadheight;
/// <summary>
///         מלבן שמטרתו לצייר את הדמות בגודל מתאים
/// </summary>
protected Rectangle rectangle;
/// <summary>
///         הזמן שבו הדמות לא מקבלת פגיעות
/// </summary>
protected Timer vulnerableTimer;
/// <summary>
///         מספר הרעידות כמה זמן
/// </summary>
```

- Defense
- direction
- downheight
- downleft
- downright
- downwidth
- Health
- invulnerableco...
- isDEAD
- isdeadheight
- isdeadleft
- isdeadright
- isdeadwidth
- ishurtheight
- ishurtleft
- ishurtright
- ishurtwidth
- isVulnerable
- jumpheight
- jumpleft
- jumplimit
- jumpright
- jumpwidth
- Mana
- maxAttack
- maxDefense
- maxHealth
- maxMana
- minAttack
- minDefense
- move1manacost
- move2manacost
- move3manacost
- percentageHea...
- percentageMana
- rectangle
- shakingTimer

```
protected Timer shakingTimer;
/// <summary>
///         איוונט לסיום המשחק כאשר נגמר לשחק נקודות החיים
/// </summary>
public event EventHandler EndGame;
/// <summary>
///         משתנים הקשורים לנקודות החיים של הדמות
/// </summary>
protected double Health, maxHealth,
                percentageHealth;
/// <summary>
///         משתנים הקשורים לנקודות המאנה של הדמות
/// </summary>
protected double Mana, maxMana, percentageMana;
/// <summary>
///         משתנים הקשורים לערך התקיפה של הדמות
/// </summary>
protected double Attack, maxAttack, minAttack;
/// <summary>
///         מתשנים הקשורים לערך ההגנה של הדמות
/// </summary>
protected double Defense, maxDefense, minDefense;
/// <summary>
///         מספר העלות במאנה לכל מתקפת
/// </summary>
protected int move1manacost, move2manacost,
                move3manacost;
/// <summary>
///         תמונה של פגיעה
/// </summary>
protected Image ishurtleft, ishurtright;
/// <summary>
///         מערכי תמונות של הדמות למטה
/// </summary>
protected Image downleft, downright;
/// <summary>
///         תמונת הדמות בקפיצה
/// </summary>
protected Image jumpleft, jumpright;
/// <summary>
///         מערכי תמונות בעמידה ישרה
/// </summary>
protected Image[] standleft, standright;
/// <summary>
///         מערכי תמונות בהליכה
/// </summary>
protected Image[] walkleft, walkright;
/// <summary>
///         מערכי תמונות ביריה
/// </summary>
protected Image[] shootleft, shootright;
/// <summary>
///         מערכי תמונות במוות
/// </summary>
protected Image[] isdeadleft, isdeadright;
/// <summary>
///         איוונט למחיקת הדמות
/// </summary>
public event EventHandler DeleteCharacter;
/// <summary>
///         טיימר להחלפת תמונות
/// </summary>
```




```
protected Timer changepicturesTimer;
/// <summary>
/// מונה את התמונה העכשוית
/// </summary>
protected int currentpictureindex;
/// <summary>
/// טיימר תזוזה בציר האיקס
/// </summary>
protected Timer velocityXTimer;
/// <summary>
/// תאוצה בציר האיקס
/// </summary>
protected int accelerationX;
/// <summary>
/// טיימר לתזוזה בציר הוואי
/// </summary>
protected Timer velocityYTimer;
/// <summary>
/// תאוצה בציר הוואי
/// </summary>
protected int accelerationY;

/// <summary>
/// פעולה בונה
/// </summary>
/// <param name="ty">סוג הדמות</param>
public Character(CharacterType ty)

/// <summary>
/// פעולה המשנה את ערך הוואי של הדמות לפי תאוצתה
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
void velocityYTimer_Tick(object sender, EventArgs e)

/// <summary>
/// פעולה המשנה את ערך איקס של הדמות לפי תאוצתה
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
public virtual void velocityXTimer_Tick(object
                                        sender, EventArgs e)
/// <summary>
/// פעולת שינוי המונה של התמונה העכשוית
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
void changepicturesTimer_Tick(object sender,
                                        EventArgs e)

/// <summary>
/// פעולת המשנה את המונה של מספר הפעמים שהדמות תהיה בלתי נראית לשנייה
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
void shakingTimer_Tick(object sender, EventArgs e)

/// <summary>
/// פעולה שעוצרת את החסינות של הדמות לאחר זמן מסויים
/// </summary>
```

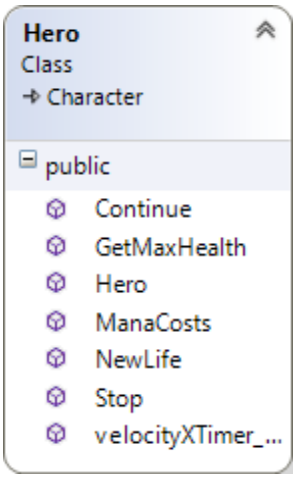
```
/// <param name="sender"></param>
/// <param name="e"></param>
void vulnerableTimer_Tick(object sender, EventArgs e)

/// <summary>
/// פעולה המציירת את הדמות
/// </summary>
/// <param name="e">עצם גרפי של האלמנט עליו אני מצייר</param>
public void ShowMe(PaintEventArgs e)

/// <summary>
/// פעולה העוצרת את הדמות
/// </summary>
public virtual void Stop()

/// <summary>
/// פעולה הממשיכה את הדמות
/// </summary>
public virtual void Continue()

/// <summary>
/// פעולה המזיזה את הדמות בלחיצה על כפתור
/// </summary>
public void Move()

/// <summary>
/// פעולה המקפיצה את הדמות .מותרת רק קפיצה אחת
/// </summary>
internal void Jump()

/// <summary>
/// פעולת התחלת החסינות מפני פגעים
/// </summary>
public void StartInvulnerability()

/// <summary>
/// מחזירה את ערך ההתקפה לערך הרגיל
/// </summary>
public void ReturnAttackToNorm()

/// <summary>
/// מחזירה את ערך ההגנה לערך הרגיל
/// </summary>
public void ReturnDefenseToNorm()

/// <summary>
/// מגבירה את ערך ההתקפה של הדמות אחוזית
/// </summary>
/// <param name="boost">אחוז ההגברה של ערך ההתקפה</param>
public void AddAttack(double boost)

/// <summary>
/// מגבירה את ערך ההגנה של הדמות אחוזית
/// </summary>
/// <param name="boost">אחוז ההגברה של ערך ההגנה</param>
public void AddDefense(double boost)
```

```
/// <summary>
///         מחזירה נקודות חיים לדמות
/// </summary>
/// <param name="heal">         מספר נקודות החיים המוחזרות</param>
public void HealHealth(int heal)
/// <summary>
///         מחזירה נקודות מאנה לדמות
/// </summary>
/// <param name="heal">         מספר  נקודות המאנה המוחזות</param>
public void HealMana(int heal)

/// <summary>
///         מחזירה את ערך ההתקפה
/// </summary>
/// <returns></returns>
public double GetAttack()
/// <summary>
///         פעולה המחסירה נוקודות חיים
/// </summary>
/// <param name="dmg">         מספר נקודות החיים להחסרה</param>
/// <param name="truedmg">         משתנה המורה אם להתייחס לערך ההגנה או לא</param>
public virtual void LooseHealth(int dmg,bool truedmg)

/// <summary>
///         מחזיר את אחוז החיים למטרת הצגה במשחק
/// </summary>
/// <returns>         מחזיר את אחוז החיים של הדמות</returns>
public double GetPercentageHealth()

/// <summary>
///         מחזיר את אחוז המאנה למטרת הצגה במשחק
/// </summary>
/// <returns>         מחזיר את אחוז המאנה של הדמות</returns>
public double GetPercentageMana()

/// <summary>
///         האם הדמות חסינה לפגיעות
/// </summary>
/// <returns>         מחזיר משתנה בוליאני</returns>
public bool GetVulnerability()

/// <summary>
///         מחזיר את סוג הדמות
/// </summary>
/// <returns></returns>
public CharacterType GetType()

/// <summary>
///         מחזיר את הכיוון של הדמות
/// </summary>
/// <returns>         תו המציג את הכיוון ימין או שמאל</returns>
public char GetDirection()

/// <summary>
///         מחזיר את מלבן הדמות
/// </summary>
/// <returns>         מלבן הדמות</returns>
internal Rectangle GetRectangle()
```

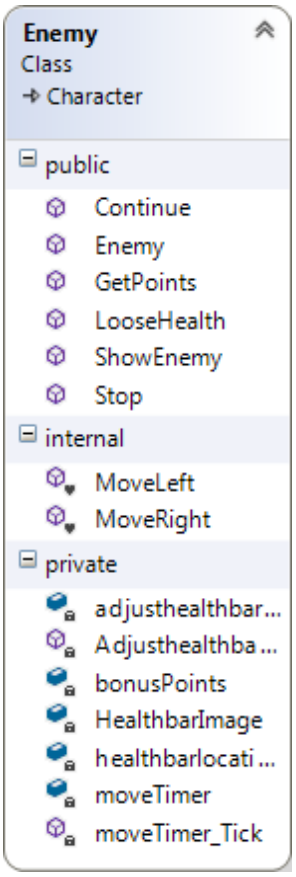
```
/// <summary>
///         מחזיר את תאוצת הדמות למטרת הזזת המסך
/// </summary>
/// <returns>         מספר המייצג את תאוצת הדמות</returns>
internal int GetAccelerationX()

/// <summary>
///         מחזיר את הגובה של הריצפה לדמות
/// </summary>
/// <param name="newfloor">         מקבל את הגובה החדש של הרצפה</param>
internal void SetArenaFloor(int newfloor)

/// <summary>
///         פעולת Get
/// </summary>
/// <returns>         האיקס השמאלי של הדמות</returns>
internal int GetCharacterX1()

/// <summary>
///         פעולת Get
/// </summary>
/// <returns>         האיקס הימני של הדמות</returns>
internal int GetCharacterX2()

/// <summary>
///         פעולת Get
/// </summary>
/// <returns>         מחזיר את גובה הוואי התחתון של הדמות</returns>
internal int GetCharacterY()

/// <summary>
///         פעולת Set
/// </summary>
/// <param name="di">         משנה את כיוון הדמות</param>
public void SetDirection(DirectionType di)

/// <summary>
///         מאפס את האינדקס של התמונה
/// </summary>
public void SetPictureIndex()

/// <summary>
///         משנה את כיוון הדמות לירייה
/// </summary>
public void SetDirectionToShoot()

/// <summary>
///         משנה את כיוון הדמות עמידה
/// </summary>
public void SetDirectionToStand()

/// <summary>
///         מאפס את המהירות של הדמות
/// </summary>
internal void SetVelocityX()

/// <summary>
///         משנה את כיוון הדמות למטה
```

```
/// </summary>
internal void SetDirectionDown()
```

Hero



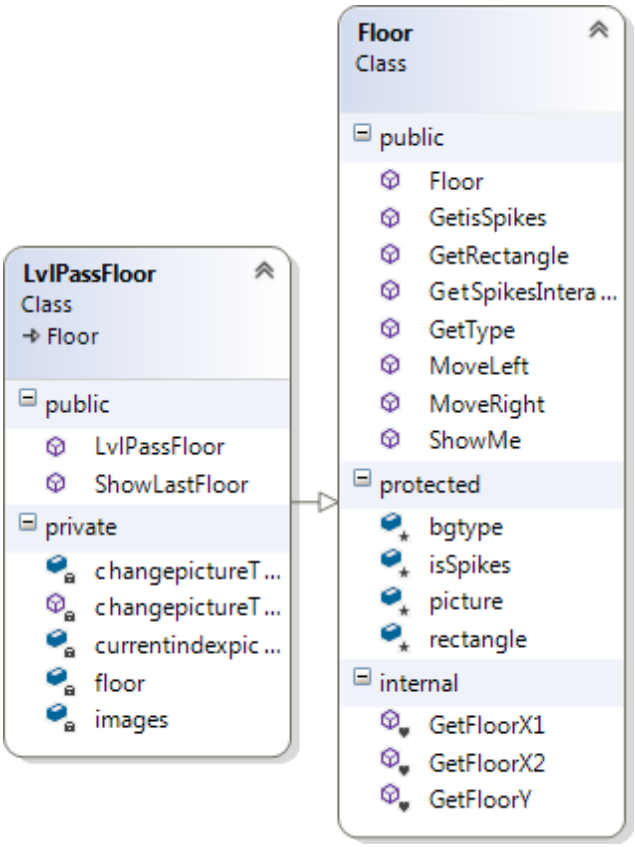

```
/// <summary>
///         פעולה בונה
/// </summary>
/// <param name="ty">        סוג הגיבור</param>
public Hero(CharacterType ty): base(ty)
/// <summary>
///         פעולת הזזת הדמות בהתאם למהירות ולכיוונה
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
public override void velocityXTimer_Tick(object
sender, EventArgs e)
//דרושה פעולת טיימר חדשה בגלל הגבולות של הדמות על המסך
/// <summary>
///פעולת עצירה
/// </summary>
public override void Stop()

/// <summary>
///         פעולת המשך
/// </summary>
public override void Continue()

/// <summary>
///         פעולה המעדכנת את מיקום הדמות במעבר שלב
/// </summary>
public void NewLife()

/// <summary>
///         פעולה הבודקת ומבצעת את המתקפה שנבחרה ע"י השחקן
/// </summary>
/// <param name="indicator">        מספר המתקפה לתקיפה</param>
/// <returns>        מחזיר שקר אם לא בוצעה המתקפה ואחרת אם היא בוצעה</returns>
public bool ManaCosts(int indicator)

//שולח את מספר המתקפה ומחזיר true אם ניתן לתקוף

/// <summary>
///         מחזיר את מספר החיים המקסימלי
/// </summary>
/// <returns>        ערך החיים המקסימלי</returns>
public int GetMaxHealth()
```



Enemy

```
/// <summary>
///         טיימר הזזת הדמות באופן מחזורי פשוט
/// </summary>
private Timer moveTimer;
/// <summary>
///         טיימר המראה את אחוז החיים של האויב
/// </summary>
private Timer adjusthealthbarTimer;
/// <summary>
///         נקודת מיקום לשורת חיים
/// </summary>
private Point healthbarlocation;
/// <summary>
///         מערך תמונות מצבים שונים של שורת החיים
/// </summary>
private Image[] HealthbarImage;
/// <summary>
///         מספר הנקודות שמביא כל מפלצת
/// </summary>
private int bonusPoints;

/// <summary>
///         פעולה בונה
/// </summary>
/// <param name="x">         מיקום איקס התחלתי</param>
/// <param name="y">         מיקום וואי התחלתי</param>
/// <param name="ty">         סוג המפלצת</param>
public Enemy(int x, int y, CharacterType ty)
    : base(ty)

/// <summary>
///         פעולת הורדת מספר נקודות חיים
/// </summary>
/// <param name="dmg">         מספר הנקודות להוריד</param>
/// <param name="truedmg">         משתנה הקובע האם להתייחס להגנה של הדמות</param>
public override void LooseHealth(int dmg, bool truedmg)

/// <summary>
///         פעולה המתאימה את מיקום השורת חיים למפלצת
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void AdjusthealthbarTimer_Tick(object sender, EventArgs e)

/// <summary>
///         פעולה המציית את האויב על המסך
/// </summary>
/// <param name="e"></param>
public void ShowEnemy(PaintEventArgs e)

/// <summary>
///         טיימר הגורם לתנועה מחזורית של המפלצות
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
void moveTimer_Tick(object sender, EventArgs e)
```

```
//טיימר שגורם לתנועה מחזורית מימין לשמאל

/// <summary>
///  פעולה המזיזה את הדמות
/// </summary>
/// <param name="p">  מספר הפיקסלים שבו נזיז את הדמות שמאלה</param>
internal void MoveLeft(int p)

//כדי שהדמות תוכל לזוז חלק ביחד עם מסך הרקע

/// <summary>
///  פעולה המזיזה את הדמות
/// </summary>
/// <param name="p">  מספר הפיקסלים שבו נזיז את הדמות ימינה</param>
internal void MoveRight(int p)

//כדי שהדמות תוכל לזוז חלק ביחד עם מסך הרקע

/// <summary>
///  פעולת עצור
/// </summary>
public override void Stop()

/// <summary>
///  פעולת המשך
/// </summary>
public override void Continue()

/// <summary>
///  פעולתGet
/// </summary>
/// <returns>  מחזיר את השווי של המפלצת
</returns>בנקודות
public int GetPoints()
```



Floor

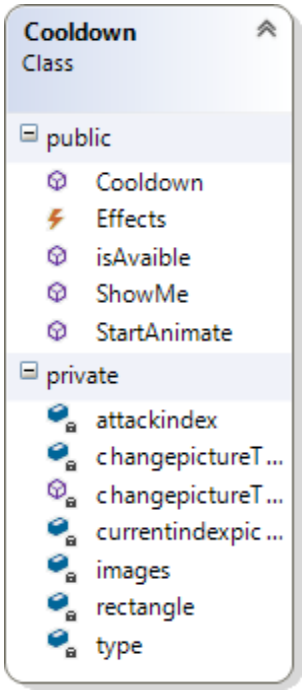
```
/// <summary>
///  מכיל את תמונת הריצפה
/// </summary>
protected Image picture;
/// <summary>
///  מלבן התמונה
/// </summary>
protected Rectangle rectangle;
/// <summary>
///  סוג הריצפה
/// </summary>
protected BackgroundType bgtype;
/// <summary>
///  האם הריצפה היא קוצים או לא
/// </summary>
protected bool isSpikes;
```

```
/// <summary>
///  פעולה בונה
/// </summary>
/// <param name="x">  איקס התחלתי של הריצפה</param>
/// <param name="y">  וואי התחלתי של הריצפה</param>
/// <param name="ty">  סוג הריצפה</param>
/// <param name="isSpikes">  אם ריצפה רוצית או לא</param>
public Floor(int x, int y,BackgroundType ty,bool isSpikes)

/// <summary>
///פעולה שמציירת את הריצפה
/// </summary>
/// <param name="e"></param> public void ShowMe (PaintEventArgs e)
/// <summary>
///  פעולתGet
/// </summary>
/// <returns>  מחזיר את המלבן</returns>
public Rectangle GetRectangle()

/// <summary>
///  פעולהGet
/// </summary>
/// <returns>  מחזיר את סוג הריצפה</returns>
public BackgroundType GetType()

/// <summary>
///  מחזיר מלבן חדש יותר קטן המתאים לקוצים
/// </summary>
/// <returns>  מחזיר מלבן של הקוצים</returns>
public Rectangle GetSpikesInteractRectangle()

/// <summary>
///  פעולהGet
/// </summary>
/// <returns>  האם הריצפה קוצית או לא</returns>
public bool GetisSpikes()

/// <summary>
///  האיקס השמאלי של הריצה
/// </summary>
/// <returns>  ערך איקס שמאלי של דמות</returns>
internal int GetFloorX1()

/// <summary>
///  האיקס הימני
/// </summary>
/// <returns>  ערך האיקס הימני של הדמות</returns>
internal int GetFloorX2()

/// <summary>
///  מחזיר את גובה הריצפה
/// </summary>
/// <returns>  ערך הוואי העליון של הריצפה</returns>
internal int GetFloorY()

/// <summary>
///  פעולת הזזה שמאלה
/// </summary>
/// <param name="p">  מספר הפיקסלים להזזה</param>
```

```
public void MoveLeft(int p)

/// <summary>
///         פעולת הזזה ימינה
/// </summary>
/// <param name="p">         מספר הפיקסלים להזזה</param>
public void MoveRight(int p)
```

L vlPassFloor

```
/// <summary>
///         ריצפה
/// </summary>
private Floor floor;
/// <summary>
///         מערך תמונות המעבר שלב
/// </summary>
private Image[] images
/// <summary>
///         טיימר החלפת תמונות
/// </summary>
private Timer changepictureTimer;
/// <summary>
///         אינדקס לתמונה העכשוית
/// </summary>
private int currentindexpicture;

/// <summary>
///         פעולה בונה
/// </summary>
/// <param name="f">   מקבל את הריצפה במיקום הכי רחוק כדי לשים שם את מעבר השלב</param>
public LvlPassFloor(Floor f)
: base(f.GetFloorX1(), f.GetFloorY(), f.GetType(), false)

/// <summary>
///         פעולה שינוי אינדקס התמונה
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
void changepictureTimer_Tick(object sender, EventArgs e)
```

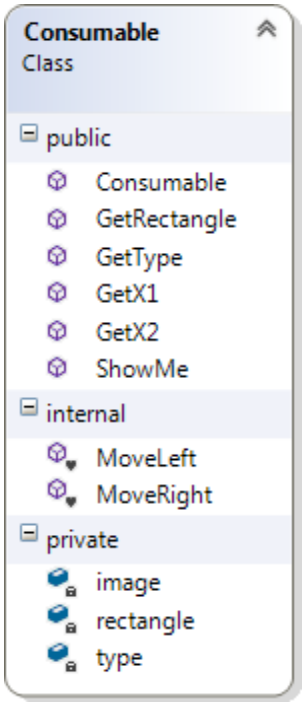


Cooldown

```
/// <summary>
/// מלבן העצם
/// </summary>
private Rectangle rectangle;
/// <summary>
/// מערך תמונות העצם
/// </summary>
private Image[] images;
/// <summary>
/// אינדקס תמונות
/// </summary>
private int currentindexpicture;
/// <summary>
/// סוג הדמות שלה מתאים העצם
/// </summary>
private CharacterType type;
/// <summary>
/// מספר המתקפה של הדמות
/// </summary>
private int attackindex;
/// <summary>
/// טיימר החלפת תמונות
/// </summary>
private Timer changepictureTimer;
/// <summary>
/// איוונט המראה למנאג'ר מה לעשות
/// </summary>
public event EventHandler Effects;

/// <summary>
/// פעולה בונה
/// </summary>
/// <param name="ty">סוג הדמות שלה מתאים עצם מסוים</param>
/// <param name="i">מספר המתאים למתקפה</param>
public Cooldown(CharacterType ty, int i)

/// <summary>
/// טיימר החלפת תמונו
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void changepictureTimer_Tick(object sender, EventArgs e)

/// <summary>
/// פעולה עוצרת
/// </summary>
public void Stop()

/// <summary>
/// פעולה ממשיכה
/// </summary>
public void Continue()

/// <summary>
/// פעולה שמציית את העצם
/// </summary>
/// <param name="e">עצם גרפי של האלמנט עליו אני מצייר</param>
```

```
public void ShowMe(PaintEventArgs e)

/// <summary>
/// התחל את האנימצייה של העצם
/// </summary>
public void StartAnimate()

/// <summary>
/// בודק אם ניתן להשתמש במתקפה שוב
/// </summary>
/// <returns></returns>
public bool isAvaible()
```

Consumable

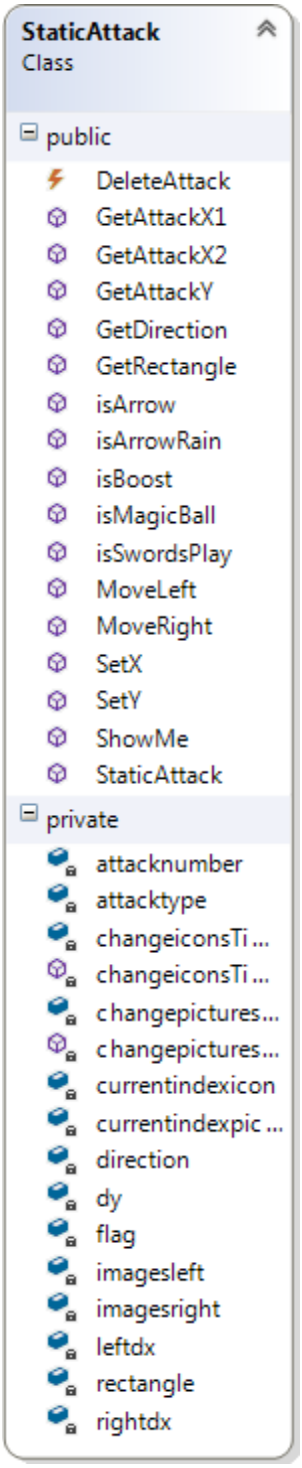



```
/// <summary>
/// התמונה של המצרך
/// </summary>
private Image image;
/// <summary>
/// המלבן של המצרך שבו נצייר את התמונה
/// </summary>
private Rectangle rectangle;
/// <summary>
/// סוג המצרך
/// </summary>
private ConsumableType type;

/// <summary>
/// פעולה בונה
/// </summary>
/// <param name="x">ערך האיקס של המצרך</param>
/// <param name="y">ערך הוואי של המצרך</param>
/// <param name="ty">סוג המצרך</param>
public Consumable(int x,int y,ConsumableType
ty)

/// <summary>
/// פעולה שמציירת את המצרך
/// </summary>
/// <param name="e">עצם גרפי של האלמנט עליו אני מצייר</param>
public void ShowMe(PaintEventArgs e)

/// <summary>
/// פעולה שמחזירה את סוג המצרך
/// </summary>
/// <returns>סוג המצרך</returns>
public ConsumableType GetType()

/// <summary>
/// פעולתget
/// </summary>
/// <returns>ערך האיקס השמאלי</returns>
public int GetX1()

/// <summary>
/// פעולתget
```

```
/// </summary>
/// <returns> ערך האיקס הימני </returns>
public int GetX2()

/// <summary>
/// פעולת Get
/// </summary>
/// <returns> ערך המלבן של המצרך </returns>
public Rectangle GetRectangle()

/// <summary>
/// פעולה שמזיזה את המצרך
/// </summary>
/// <param name="p"> מספר הפיקסלים שנזיז את המצרך שמאל </param>
internal void MoveLeft(int p)

/// <summary>
/// פעולה שמזיזה את המצרך
/// </summary>
/// <param name="p"> מספר הפיקסלים שנזיז את המצרך ימינה </param>
internal void MoveRight(int p)
```

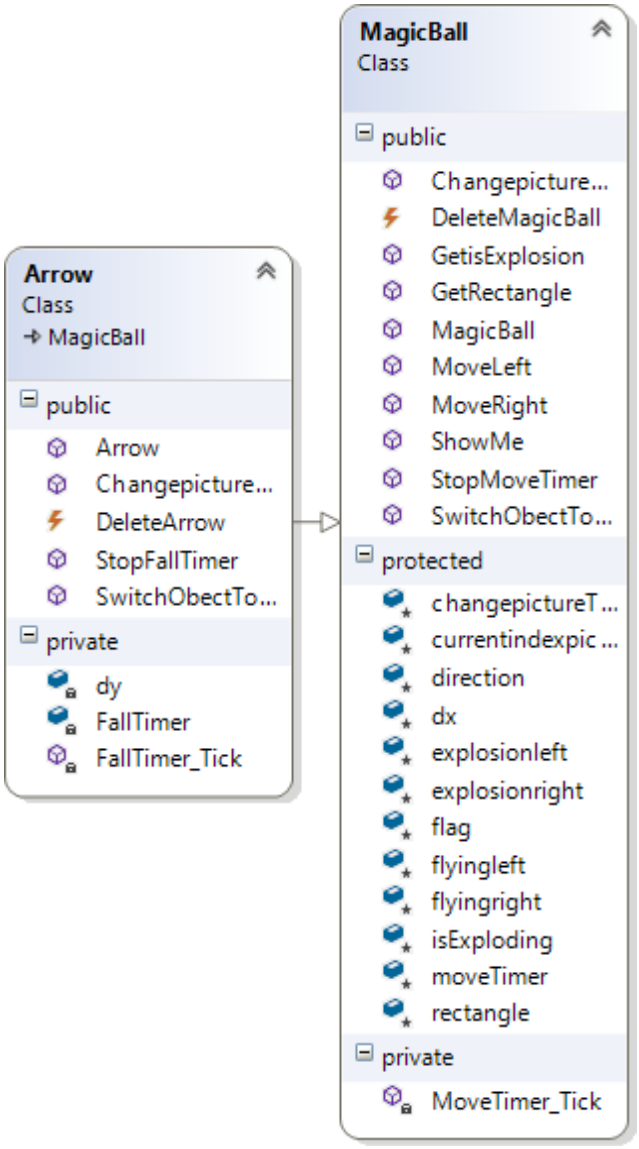
StaticAttack

```
/// <summary>
/// מלבן המתקפת
/// </summary>
private Rectangle rectangle;
/// <summary>
/// מערך תמונות ימינה ושמאלה
/// </summary>
private Image[] imagesright, imagesleft;
/// <summary>
/// אינדקס תמונה עכשוית
/// </summary>
private int currentindexpicture;
/// <summary>
/// טיימר החלפת תמונות
/// </summary>
private Timer changepicturesTimer;
/// <summary>
/// תו המראה את כיוון המתקפה
/// </summary>
private char direction;
/// <summary>
/// מרחק המתקפה מהדמות משני הצדדים
/// </summary>
private int leftdx, rightdx, dy;
/// <summary>
/// " משתנה שמטרתו להפסיק את פעולת העצם אם העצם" מת
/// </summary>
private bool flag;
/// <summary>
/// איוונט מחיקת העצם
/// </summary>
public event EventHandler DeleteAttack;
/// <summary>
```




```
                                                    /// סוג המתקפה המתאימה לגיבור
                                                    /// </summary>
                                                    private CharacterType attacktype;
                                                    /// <summary>
                                                    /// מספר המתקפה
                                                    /// </summary>
                                                    private int attacknumber;

                                                    /// <summary>
                                                    /// פעולה בונה
                                                    /// </summary>
                                                    /// <param name="x">איקס התחלתי</param>
                                                    /// <param name="y">וואי התחלתי</param>
                                                    /// <param name="attnum">מספר המתקפה</param>
                                                    /// <param name="ty">סוג המתקפה</param>
                                                    /// <param name="di">כיוון המתקפה</param>
                                                    public StaticAttack(int x,int y,int
                                                    attnum,CharacterType ty,char di)

                                                    /// <summary>
                                                    /// החלפת אינדקס תמונות
                                                    /// </summary>
                                                    /// <param name="sender"></param>
                                                    /// <param name="e"></param>
void changepicturesTimer_Tick(object sender, EventArgs
                                                    e)

                                                    /// <summary>
                                                    /// פעולה המציירת את המתקפה
                                                    /// </summary>
                                                    /// <param name="e">></param>
                                                    public void ShowMe(PaintEventArgs e)

                                                    /// <summary>
                                                    /// פעולה המתאימה את האיקס של המתקפה לפי הדמות
                                                    /// </summary>
                                                    /// <param name="x">ערך האיקס של הדמות</param>
                                                    public void SetX(int x)

                                                    /// <summary>
                                                    /// פעולה המתאימה את הוואי של המתקפה לפי הדמות
                                                    /// </summary>
                                                    /// <param name="y">ערך הוואי של הדמות</param>
                                                    public void SetY(int y)

                                                    /// <summary>
                                                    /// בודק אם המתקפה היא עושה פגיעות או לא
                                                    /// </summary>
                                                    /// <returns>מחזיר נכון אם היא לא פוגעת ולא נכון אם היא פוגעת</returns>
                                                    public bool isBoost()

                                                    /// <summary>
                                                    /// פעולת הזזת המתקפה שמאלה
                                                    /// </summary>
                                                    /// <param name="p">מספר הפיקסלים להזזה</param>
                                                    public void MoveLeft(int p)

                                                    /// <summary>
```

```
/// פעולת הזזת המתקפה ימינה
/// </summary>
/// <param name="p">מספר הפיקסלים להזזה</param>
public void MoveRight(int p)

/// <summary>
/// בודק אם סוג המתקפה היא חץ
/// </summary>
/// <returns>מחזיר נכון אם כן, אם לא אחרת</returns>
public bool isArrow()

/// <summary>
/// בודק אם המתקפה מסוג כדור חשמל
/// </summary>
/// <returns>מחזיר נכון אם כן אחרת אם לא</returns>
public bool isMagicBall()

/// <summary>
/// בודק אם המתקפה מסוג משחק חרבות
/// </summary>
/// <returns>מחזירה נכון אם כן אחרת אם לא</returns>
public bool isSwordsPlay()

/// <summary>
/// בודק אם המתקפה מסוג גדם חצים
/// </summary>
/// <returns>מחזירה נכון אם כן אחרת אם לא</returns>
public bool isArrowRain()

/// <summary>
/// פעולת Get
/// </summary>
/// <returns>מחזיר תו המצין כיוון</returns>
public char GetDirection()

/// <summary>
/// פעולת Get
/// </summary>
/// <returns>מחזיר את האיקס הימני</returns>
public int GetAttackX2()

/// <summary>
/// פעולת Get
/// </summary>
/// <returns>מחזיר את האיקס השמאלי</returns>
public int GetAttackX1()

/// <summary>
/// פעולת Get
/// </summary>
/// <returns>מחזיר את הוואי העליון</returns>
public int GetAttackY()

/// <summary>
/// פעולת Get
/// </summary>
/// <returns>מחזיר את המלבן של המתקפה</returns>
public Rectangle GetRectangle()
```



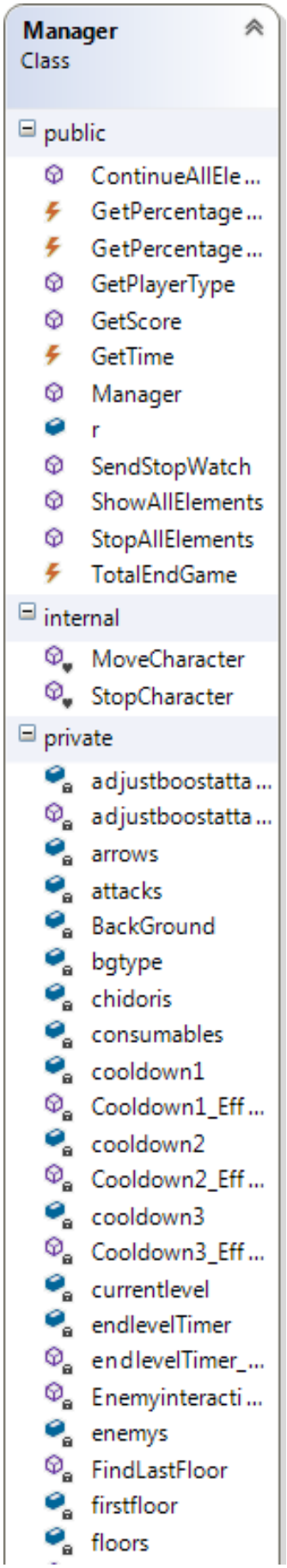
MagicBall

```
/// <summary>
///         מערך תמונות בתעופה
/// </summary>
protected Image[]
flyingleft,flyingright;
/// <summary>
///         מערך תמונות בפיצוץ
/// </summary>
protected Image[] explosionleft,
explosionright;
/// <summary>
///         מלבן המתקפה
/// </summary>
protected Rectangle rectangle;
/// <summary>
///         טיימר שינוי המקום של המתקפה
/// </summary>
protected Timer moveTimer;
/// <summary>
///         מספר הפיקסלי להזזה בציר האיקס
/// </summary>
protected int dx;
/// <summary>
///         טיימר החלפת תמונות
/// </summary>
protected Timer
changepictureTimer;
/// <summary>
///         אינדקס תמונה עכשוית
/// </summary>
protected int
currentindexpicture;
/// <summary>
///         תו המראה על כיוון
/// </summary>
protected char direction;
/// <summary>
///         פעולה הבודקת אם המתקפה מפוצצת
/// </summary>
protected bool isExploding;
/// <summary>
/// "         משתנה שנועד להפסיק את פעולת העצם אם הוא" מת
/// </summary>
protected bool flag;
/// <summary>
///         איוונט מחיקת המתקפה
/// </summary>
public event EventHandler DeleteMagicBall;

/// <summary>
///         פעולה בונה
/// </summary>
/// <param name="x">         האיקס ההתחלתי</param>
/// <param name="y">         הוואי ההתחלתי</param>
/// <param name="c">         תו המייצג את כיוון הדמות</param>
public MagicBall(int x, int y, char c)

/// <summary>
```

```
/// פעולת שינוי האינדקס התמונות העכשווי
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
public virtual void ChangepictureTimer_Tick(object sender, EventArgs e)

/// <summary>
/// טיימר הזזת הדמות
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void MoveTimer_Tick(object sender, EventArgs e)

/// <summary>
/// פעולה המציירת את הדמות
/// </summary>
/// <param name="e"></param>
public void ShowMe(PaintEventArgs e)

/// <summary>
/// פעולה המחליפה את המתקפה למצב פיצוץ
/// </summary>
public virtual void SwitchObectToExplosion()

/// <summary>
/// פעולתGet
/// </summary>
/// <returns> מחזיר מלבן</returns>
public Rectangle GetRectangle()

/// <summary>
/// פעולת הזזה שמאלה
/// </summary>
/// <param name="p">מספר הפיקסלים שבו יש להזיז את המתקפה</param>
public void MoveLeft(int p)

/// <summary>
/// פעולת הזזה ימינה
/// </summary>
/// <param name="p">מספר הפיקסלים שבו יש להזיז את המתקפה</param>
public void MoveRight(int p)

/// <summary>
/// פעולתGet
/// </summary>
/// <returns> נכון אם הדמות בפיצוץ לא נכון אחרת</returns>
public bool GetisExplosion()

/// <summary>
/// פעולת עצירת המתקפה
/// </summary>
public void StopMoveTimer()
```

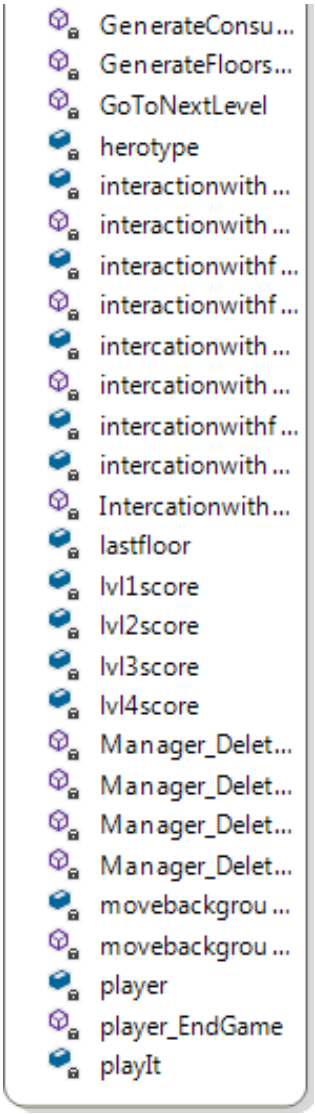
Arrow

```
/// <summary>
///         טיימר נפילה
/// </summary>
private Timer FallTimer;
/// <summary>
///         מספר הפיקסלים שבו ניפול בכלTick
/// </summary>
private int dy;
/// <summary>
///         אייונט המוחק את העצם
/// </summary>
public event EventHandler DeleteArrow;

/// <summary>
///         פעולה בונה
/// </summary>
/// <param name="x">        ערך האיקס ההתחלתי של החץ</param>
/// <param name="y">        ערך הוואי ההתחלתי של החץ</param>
/// <param name="c">        תו המראה על כיוון החץ .ימינה או שמאלה</param>

public Arrow(int x, int y,char c):base(x,y,c)


/// <summary>
///         פעולה המשנה את מונה התמונות העכשוי
/// </summary>
/// <param name="sender">></param>
/// <param name="e">></param>
public override void ChangepictureTimer_Tick(object sender, EventArgs
e)

/// <summary>
///         פעולה הגורמת לנפילת החץ
/// </summary>
/// <param name="sender">></param>
/// <param name="e">></param>
private void FallTimer_Tick(object sender, EventArgs e)

/// <summary>
///         פעולה שמחליפה את מערך התמונות לפיצוץ
/// </summary>
public override void SwitchObectToExplosion()

/// <summary>
///         עצור את הנפילה של העצם
/// </summary>
public void StopFallTimer()
```



Manager

```
//-----------------------------------------------------------------

                                        /// <summary>
                        דלגייט המשמ לעדכון שורות החיים והמאנה    ///
                                        /// </summary>
                    /// <typeparam name="T"></typeparam>
                            /// <param name="t"></param>
                public delegate void SendLoses<T>(T t);
                                        /// <summary>
                      דלגייט המשמש לשליחת הניקוד בסוף משחק    ///
                                        /// </summary>
                    /// <typeparam name="T"></typeparam>
                          /// <param name="sc1"></param>
                          /// <param name="sc2"></param>
                          /// <param name="sc3"></param>
                          /// <param name="sc4"></param>
public delegate void SendScores<T>(T sc1, T sc2, T sc3,
                                                T sc4);

//-----------------------------------------------------------------

                                        /// <summary>
                                    הגיבור        ///
                                        /// </summary>
                                private Hero player;
                                        /// <summary>
                                  סוג הגיבור        ///
                                        /// </summary>
                        private CharacterType herotype;
                                        /// <summary>
                                    האויבים        ///
                                        /// </summary>
                            private List<Enemy> enemys;
                                        /// <summary>
                                     האוכל        ///
                                        /// </summary>
                    private List<Consumable> consumables;
                                        /// <summary>
                                    הריצפות        ///
                                        /// </summary>
                            private List<Floor> floors;
                                        /// <summary>
                                הריצפה הראשונה        ///
                                        /// </summary>
                              private Floor firstfloor;
                                        /// <summary>
                                הריצפה האחרונה        ///
                                        /// </summary>
                        private LvlPassFloor lastfloor;
                                        /// <summary>
                                  ההתקפות הסטטיות        ///
                                        /// </summary>
                     private List<StaticAttack> attacks;
                                        /// <summary>
                                 הכדורי קסם/ברק        ///
                                        /// </summary>
```



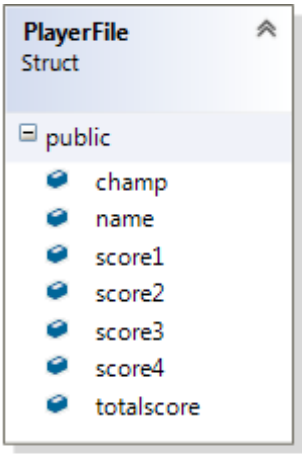

```
private List<MagicBall> chidoris;
/// <summary>
/// החצים
/// </summary>
private List<Arrow> arrows;
/// <summary>
/// הרקע של המשחק
/// </summary>
private Image BackGround;
/// <summary>
/// האייקונים של המתקפות וזמן המחזור שלהם
/// </summary>
private Cooldown cooldown1, cooldown2, cooldown3;
/// <summary>
/// נגן מוסיקה
/// </summary>
private SoundPlayer playIt;

/// <summary>
/// אינדקס לרמה
/// </summary>
private int currentlevel;
/// <summary>
/// משתנה רנדומלי
/// </summary>
public static Random r;
/// <summary>
/// סוג הרקע
/// </summary>
private BackgroundType bgtype;
/// <summary>
/// טיימר התנגשות עם ריצפה
/// </summary>
private Timer interactionwithfloorTimer;
/// <summary>
/// טיימר הזזת רקע
/// </summary>
private Timer movebackgroundTimer;
/// <summary>
/// טיימר התנגשות עם אוכל
/// </summary>
private Timer interactionwithconsumableTimer;
/// <summary>
/// טיימר התנגשות עם אויבים
/// </summary>
private Timer intercationwithenemyTimer;
/// <summary>
/// טיימר התאמת המתקפת לדמות
/// </summary>
private Timer adjustboostattacks;
/// <summary>
/// טיימר התנגשות אויבים עם מתקפות עפות
/// </summary>
private Timer intercationwithflyingattack;
/// <summary>
/// טיימר התנגשות עם מתקפה סטטית
/// </summary>
private Timer intercationwithstaticattack;
/// <summary>
/// טיימר בדיקת התנגשות עם מעבר לשלב הבא
/// </summary>
private Timer endlevelTimer;
```

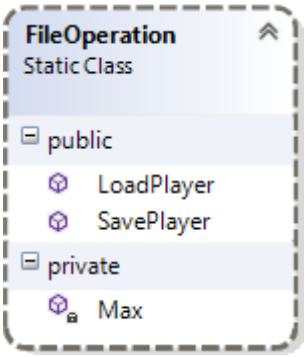
```
/// <summary>
///         הניקוד לכל רמת משחק
/// </summary>
private int lvl1score, lvl2score, lvl3score, lvl4score;

/// <summary>
///         שני איוונטים להתאמת שורות החיים והמאנה
/// </summary>
public event SendLoses<double> GetPercentageHealth, GetPercentageMana;
/// <summary>
///         איוונטים לעצירה והפעלת הטיימר
/// </summary>
public event EventHandler StopTime, ContinueTime;
/// <summary>
///         איוונט המוסר את הניקוד בסוף או בהפסד המשחק
/// </summary>
public event SendScores<int> TotalEndGame;
/// <summary>
///         איוונט המקבל את הזמן סיום השלב
/// </summary>
public event EventHandler GetTime;

/// <summary>
///         פעולה בונה
/// </summary>
/// <param name="ty">        סוג הדמות שנבחר</param>
public Manager(CharacterType ty)

/// <summary>
///         פעולת סיום המשחק
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
void player_EndGame(object sender, EventArgs e)

/// <summary>
///         בדיקת סיום הרמה
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void endlevelTimer_Tick(object sender, EventArgs e)

/// <summary>
///         בדיקת התנגשות עם מתקפה סטטית
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void Intercationwithstaticattack_Tick(object sender, EventArgs
e)

/// <summary>
///         בדיקת התנגשות עם מתקפה עפה
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void Intercationwithflyingattack_Tick(object sender, EventArgs
e)
```

```
/// <summary>
/// מתאים את ההתקפה לדמות
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void adjustboostattacks_Tick(object sender, EventArgs e)

/// <summary>
/// פעולת הזזת הרקע
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void movebackgroundTimer_Tick(object sender, EventArgs e)

/// <summary>
/// בדיקת התנגשות הדמויות עם הריצפה
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void interactionwithfloorTimer_Tick(object sender, EventArgs e)

/// <summary>
/// פעולת התנגשות האוכל עם הגיבור
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void interactionwithconsumableTimer_Tick(object sender,
EventArgs e)

/// <summary>
/// בדיקת התנגשות הגיבור עם האויבים
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void intercationwithenemyTimer_Tick(object sender, EventArgs e)

/// <summary>
/// פעולה המציירת את כל הדמויות ועצמים
/// </summary>
/// <param name="e"></param>
public void ShowAllElements(PaintEventArgs e)

/// <summary>
/// פעולה העוצר את כל העצמים
/// </summary>
public void StopAllElements()

/// <summary>
/// פעולה הממשיכה את כל העצמים
/// </summary>
public void ContinueAllElements()

/// <summary>
/// פעולה המזיזה את הגיבור בהקשת מקשים
/// </summary>
/// <param name="keyCode">המקש שהוקש</param>
internal void MoveCharacter(Keys keyCode)
```

```
/// <summary>
/// פעולת מחיקת ההתקפה
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
void Manager_DeleteAttack(object sender, EventArgs e)

/// <summary>
/// פעולת מחיקת כדור הקסם
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void Manager_DeleteMagicBall(object sender, EventArgs e)

/// <summary>
/// פעולת מחיקת החץ
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void Manager_DeleteArrow(object sender, EventArgs e)

/// <summary>
/// פעולת מחיקת האויב
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void Manager_DeleteCharacter(object sender, EventArgs e)

/// <summary>
/// פעולת עצירת הדמות בתנועתה
/// </summary>
/// <param name="keyCode"></param>
internal void StopCharacter(Keys keyCode)

/// <summary>
/// החזרת נתוני התקפה והגנה של הגיבור לנורמה
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void Cooldown2_Effects(object sender, EventArgs e)

/// <summary>
/// החזרת נתוני הגיבור לנורמה
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void Cooldown1_Effects(object sender, EventArgs e)

/// <summary>
/// פעולת עזר ליצירת רצפות
/// </summary>
/// <param name="locationx">מיקום איקס התחלתי לשרשרת הריצפות</param>
/// <param name="locationy">מיקום וואי התחלתי לשרשרת הריצפות</param>
/// <param name="numberoffloors">מספר הריצפות</param>
private void GenerateFloors(int locationx, int locationy, int
numberoffloors)

/// <summary>
```

```
/// פעולה שנייה של עזר ליצירת ריצפות
/// </summary>
/// <param name="locationx">מיקום איקס התחלתי לשרשרת הריצפות</param>
/// <param name="locationy">מיקום וואי התחלתי לשרשרת הריצפות</param>
/// <param name="numberoffloors">מספר הריצפות</param>
/// <param name="spikey">אם הריצפה דוקרת או לא</param>
private void GenerateFloors(int locationx, int locationy, int
numberoffloors, bool spikey)

/// <summary>
/// פעולת יצירת אוכל על הריצפות
/// </summary>
private void GenerateConsumables()

/// <summary>
/// פעולת עזר שעוזרת לי למצוא את הריצפה הרחוקה ביותר
/// </summary>
/// <returns></returns>
private Floor FindLastFloor()

/// <summary>
/// פעולת עזר שמחליפה את כל הרמה לרמה הבאה
/// </summary>
private void GoToNextLevel()

/// <summary>
/// מחזיר את שם הגיבור בו השתמש השחקן
/// </summary>
/// <returns></returns>
public string GetPlayerType()

/// <summary>
/// פעולה המקבלת את הזמן בו סיים השחקן את המשחק
/// </summary>
/// <param name="stopwatch">שעון עצר</param>
public void SendStopWatch(Stopwatch stopwatch)

/// <summary>
/// פעולתGet
/// </summary>
/// <returns>מחזיר את הניקוד לרמה העכשוית</returns>
public int GetScore()
```





PlayerFile

```
/// <summary>
///     סטראקט של שם המשחקן וניקודו
/// </summary>
public struct PlayerFile
```

FileOperation

```
// <summary>
///     הפעולה מאיינת ושומרת את חמשת המוב ילים בקובץ טקסט
/// </summary>
/// <param name="name"></param>
/// <param name="score1"></param>
public static void SavePlayer(string name, int
score1,int score2,int score3,int score4,string champ)
```

```
/// <summary>
///     הפעולה טוענת את המובילים מקובץ הטקסט על מנת לבצע מיון יחד עם
תוצאות חדשות
/// </summary>
/// <returns></returns>
public static List<PlayerFile> LoadPlayer()
```

```
/// <summary>
///     הפעולה בודקת איזו תוצאה גבוה יותר מבין 2 תוצאות
/// </summary>
/// <param name="p1"></param>
/// <param name="p2"></param>
/// <returns></returns>
private static int Max(PlayerFile p1,PlayerFile p2)
```

בעיות אלגוריתמיות פתורות

במהלך כתיבת התוכנית נפגשתי עם לא מעט בעיות אלגורתמיות, הצלחתי לפתור את כולן. אפרט עתה את הקשות ביותר:

1. תנועת הדמות על הריצפה – הבעיה הייתה בלגרום לדמות לעבור מריצפה גבוהה לנמוכה ולהיפך. זה בעייתי מפני שלדמות ולרצפה יש Rectangle אך על הדמות לעלות מעל הרצפה הגבוהה כדי להיות עליה (כמו במציאות) ולכן לא ניתן להשתמש במקרה זה בפעולה r1.IntersectsWith(Rectangle r2) . לכן האלגוריתמם שאני הצעתי הוא: עבור כל ריצפה במשחק: בדוק אם איקס ימני של הדמות גדול מאיקס שמאלי של הריצפה וגם בדוק אם איקס שמאלי של הדמות קטן מאיקס ימני של הריצפה וגם בדוק אם וואי תחתון של הדמות נמצאת מעל וואי עליון של רצפה. אם כן, עדכן את ערך הריצפה של הדמות לערך הוואי של הריצפה.

   כך פתרתי את בעיית התנועה. יש לציין שעל כל מלבני הריצפה להיות בעלי אורך מאורך הדמות כדי שלגוריתם זה יפעל. זהו תמיד המקרה בפרויקט שלי לכן התנועה עובדת.

2. הצגת שורת החיים והמאנה – הבעייתיות במקרה זה הייתה שערכי החיים והמאנה מתעדכנים במחלקה Character והיא נמצאת בתוך המחלקה Manager שנמצאת בתוך הטופס GameForm שבו מוצגים השורות חיים ומאנה. לכן הפיתרון היה להשתמש באיוונטים אך לא יעיל להעביר את ה Sender בשלמותו בכל פעם שמתעדכנים החיים לכן היה צורך ב Delegate שיעביר את אחוז החיים ממחלקה מוכלת למחלקה מכילה עד ל GameForm ושם תתעדכן שורת החיים. ה Delegate שלי מעביר משתנה גנרי אחד, שהוא במקרה הזה משתנה מסוג double המכיל את אחוז החיים.
3. הצגת האוכל והמשקאות על המפה – בתוכנית שלי ישנם 3 סוגי "מאכלים" והייתי צריך להכניס אותם על המפה אך לא הגיוני שהמאכלים פשוט ירחפו באוויר במקום רנדומלי. לכן הפיתרון שלי היה להגריל מתוך רשימת הרצפות רצפה מסוימת ולהשתמש בערכי האיקס וואי שלה כדי ליצור מאכל בול עליה. כך מיקום הבונוסים ("מאכלים") הגיוני.
4. בעיות באיוונט מחיקת האויבים ובניקוד – בתוכנית שלי האויבים מתים בעקבות מתקפות השחקן. כאשר אויב מת יש למחוק אותו מהרשימה של האויבים כדי שלא ימשיך להופיע על המסך. בשביל זה למדתי להשתמש באיוונטים של מחיקה עבור כל אויב כאשר הוא מת. הבעיה היתה שבאיוונט אנו מגדירים אויב חדש שיכיל את הSender ובעזרתו נוכל למחוק את האויב מן הרשימה. הבעיה היתה שהאויב החדש שיצרנו (מתוך ה Sender) גם הוא קיים את התנאי שגורם לשימוש באיוונט מחיקה וכך בעצם איוונט המחיקה פעל

שוב ושוב בלי סוף ולכן גם הניקוד עלה בלי סוף כאילו שהשחקן הרג עוד ועוד אויבים. הפיתרון שלי היה להשתמש במשתנה בוליאני flag שיהפוך להיות false כאשר הדמות מתה וכך בעצם התנאי הדרוש לשימוש באיוונט לא יתקיים בשנית ולא נקבל ריבוי ניקוד.

בעיות אלגוריתמיות לא פתורות

התוכנית קורסת לאחר שימוש בה. אין סיבה נראית לעין ואין הודעת שגיעה הגיונית מהתוכנית שמראה על בעיה כלשהי. הקריסה נעשית בעיקר כאשר נשתמש בArcher. וזאת מפני שהוא הדמות שדורש מהתוכנית לשתמש בהכי הרבה RAM בגלל החצים שלו שנורים עד 4 פעמים בשנייה. אין לי או למורה שלי הצעה לפיתרון הבעיה. אני מציע פשוט להריץ את התוכנית על מחשב חזק יותר כדי שהיא לא תקרוס.

הצעות לשיפור המשחק

אילו היה לי עוד זמן לעבוד על הפרויק הייתי מוסיף את הדברים הבאים:

1. עוד דמויות לבחירה במשחק.
2. מוזיקת רקע ורעשים נוספים.
3. אפשרות ליצור רמות לשמור אותן.
4. בוס (אויב חזק משאר האויבים) לכמעט כל רמה.

מקורות

- הספר ללימוד #C של האתר Corner

http://www.corner.co.il/

- חלק רב מן התמונות נמצאו באמצעות מנוע החיפוש של גוגל

https://www.google.co.il/

- האתר להורדת תמונות BannedStory 4 – Maple Simulator

http://www.maplesimulator.com/node/13

- סאונד ורקע לכפתורים מהמשחק League of Legends:

http://eune.leagueoflegends.com/

- האתר למשחקים ברשת Nitrome :

http://www.nitrome.com/

- רעשים וסאונד אפקט מהאתר Youtube

https://www.youtube.com/

לסיכום

אולג אמר לנו בתחילת השנה: "מה שחשוב זה התהליך ,לא התוצר הסופי", הוא צדק. התהליך של הכנת הפרויקט ביחד עם חברים בכיתה היה מאוד מהנה ומוציא משגרת לימודים רגילה. הכתיבת קודים גרמה לנו לרצות ללמוד עוד כדי לשפר את המשחק וליצור משהו יותר טוב שהוא שלנו כי אנחנו יצרנו אותו. כל פעם שהצלחתי לרשום אלגוריתם שעובד והייתי מראה לחברים זו היתה גאווה. ואפילו כשפישלתי בגדול והראתי לחברים אנחנו פשוט צחקנו מזה. האווירה היתה מעולה, שמנו מוזיקה כדי להתרכז ושרנו ביחד על שירים שאנחנו אוהבים. בנוסף גם עזרנו אחד לשני בבעיות בקוד. הנחמדים ביותר היו אפילו מגיעים לשיעורים ועוזרים כולם ושוכחים לעבוד על הפרויקט שלהם. בזכות פרויקט הגמר למדנו עוד דברים במדעי המחשב ואני רוצה להעמיק וללמוד עוד אוניברסיטא על כך, אין משהו בצורת החשיבה שיש לגייס במקצוע זה שאני לא אוהב.

אני רוצה להגיד תודה לחברים שלי מהכיתה שעזרו לי בתכנות : אופק, גבריאל, עדן, רעות, כריסטינה, תמיר, אופיר, עמית, איילון, בר, רום, ברק,ליאור, מיכאלה. בין אם הסייוע היה בבדיקת עמידות התוכנית ,במתן עצה על הקוד או אפילו במבט שנתן לי השראה או רעיון להמשיך את הפרויק.

אני רוצה גם להגיד תודה לאולג על הלמידה ,היחס ,העידוד , והעזרה הפרטנית שנתן לכל אחד בכיתה. אני לא חושב שאפשר ללמד תכנות באופן טוב יותר ממנו.