# 

# **עבודת גמר**

# **לקבלת תואר טכנאי**

# **הנדסת תוכנה**

**נושא הפרויקט:**XCom Tactics

**שפת תכנות:** Java

**סביבת העבודה**‎Eclipse

**שם המגיש:**קלמנוביץ' פיוטר

**ת"ז:**XXXXXXXXX

**מכללה:** אורט חולון

**כיתה:** י"ג 2,מגמת הנדסת תוכנה

**מנחה:**XX XXXXX

**מאי2017 תשע"ז**

**תוכן עניינים**

**הצעתפרויקט.........................................................................................................**5-3

1. **מבוא .............................................................................................................** 6
   1. **מטרה .............................................................................................** 6
   2. **תיאורהמערכת.................................................................................**6
   3. **מבנההמסמך ..................................................................................** 6
   4. **ביבליוגרפיה.....................................................................................** 6
2. **מפרטי תוכנה .............................................................................................** 12-7
   1. **תיאור כללי ....................................................................................** 7-8
   2. **ממשק אדם למשתמש.....................................................................** 9-10
   3. **יכולות פונקציונאליות.......................................................................** 11
   4. **תיאור אלגוריתמים .........................................................................** 12
3. **תכנון.........................................................................................................** 13-37
   1. **פירוט המודולים+רשימת נתונים.....................................................** 13-36
   2. **מבנה נתונים .................................................................................** 37
4. **מדריך למשתמש.........................................................................................**38-38
   1. **הוראות התקנה ( Setup ) ...........................................................** 38
   2. **הוראות הפעלה ............................................................................**38
   3. **הודעות מערכת ..........................................................................** 38

**5. נספח - קוד מקור ...........................................................................................** 39-145

# **הצעת נושא לפרויקטים לטכנאים במגמת הנדסת תוכנה**

**א. הנושא** Xcom Tactics

**ב. שם המנחה :** XX XXXXX

**ג. שם התלמיד :**פיוטר קלמנוביץ' **ת.ז. : XXXXXXXXX**

**ד. עבודה ביחיד**

**ה. אופי עבודת הפרויקט :** תכנון ובניית אב טיפוס

**ו. מקום ביצוע הפרוייקט** **:** מכללת אורט חולון

**ז. תאור נושא המשחק :**

Xcom Tactics הוא משחק לוח המחולק למשבצות. במשחק משתתפים השחקן שמייצג את בני האדם והמחשב שמייצג את החייזרים. בתחילת המשחק מונחות על לוח המשחק 2 גופים , כל אחד מהם מייצג יחידה (אחת של בני אדם והשנייה של החייזרים). כל שחקן זז בתורו עם היחידה שלו ומנסה להשמיד את היחידה האויבת בעזרת החפצים שיש לו(רימונים , נשקים וכד).  
תזוזה תעשה על ידי מקשי הכיוון , ירי יעשה על ידי לחיצת כפתורים WASD שיקבעו כיוון ירי.   
חוקי המשחק:  
\* מהלך התור הולך כך: אפשר לזוז או לירו על ידי הכפתורים . הירי מתבצע במיקום היחידה .

\*בכל יחידה מתחילים עם 4 חיילים שנבחרו

כרגע יהיו לבחירה רק: Ranger,Heavy,Support,Sniper

הכוחות שיהיו על הלוח הן יהיו מבוססות על הקבוצת חיילים שנבחרה.

\*במשחק יהיו 3 סוגי בלוקים : קיר מלא , חצי קיר ורצפה . קיר מלא נותן מקסימום אחוזי הגנה , חצי קיר נותן רק חצי ממקסימום אחוזי הגנה ורצפה לא נותן בכלל אחוזי הגנה. אפשר לזוז רגיל על רצפה . אי אפשר שיחידה תעמוד באמצע קיר מלא או חצי קיר. לכל קיר יש נקודות חיים שיורדות עם כמות הירי שהם סופגות.כשרמת החיים מגיעה ל0 הקיר יורדת (והחיים מתאפסים) ברמתה עד לרצפה.

\*המערכת קרב במשחק הינה תהיה מבוססת על אחוזי פגיעה . ניתן להשפיע על אחוזי הפגיעה על ידי מרחק מהמטרה ומאחורי איזה בלוק נמצאת המטרה.נגיד שאחוזי פגיעה של נשק רגיל זה 75% והמטרה נמצאת מאחורי חצי קיר שמביאה להם 30 הגנה ונמצאת במרחק 10 קוביות מהיחידה שלנו. מרחק הירי האופטימלי הוא בין 5 קוביות ל8 קוביות, כל הבדל של קובייה מהגבול מוריד שני אחוז פגיעה. אז הגענו לחישוב שמחשב את אחוז הפגיעה שלנו במטרה.

75-30-(10-8)\*2 = 41%

\*הנשק לא מושלם ולכן תמיד יכולים לירו בזוויות קטנה פה ושם. בגלל זה כשהיחידה יורה , שטח הפגיעה שלה הוא בקונוס . והיריה עוברת דרך השורה שמוקצה עבורה.

\* הקירות הינם "אוכלות" את אחוזי הפגיעה של הכדורים שעפים אם ניקח את הדוגמא הקודמת שלנו עם 41% אז אם היחידה שלנו עומדת מאחורי בלוק קיר שלם אז מ 41% יורד 60% ואז זה יוצא מספר שלילי. ברגע שמספר שלילי סימן שהכדור הגיע למהירות של 0 ונעצר ולא ימשיך לקוביות הבאות.

הכדור יעצר גם במקרה ופוגע ביחידה כל שהי.

\* לכל חייל מיוחד יהיה את החבילת חפצים ומיומנויות שלו . למשל SUPPORT

מחזיק את הרובה הרגיל ,היכולת שלו לזרוק פצצת עשן שמשפיעה על איזור , כל מי שנמצא בתוך האיזור מקבל 20 הגנה גם אויבים וגם חברים.

הרובע סביר ממרק בינוני וטוב ממרחק קצר.

\*מערכת החיים של היחידה , כאשר נפגעת יחידה אחד מהחיילים המיוחדים שלה נהרג (ראנדומלי בדרך כלל) ,זאת אומרת שלא תנוהל היחידה על ידי החייל המיוחד אם תאסוף היחידה את הכח שלו .

משמעות הדבר לכל יחידה יש 4 יחידות חיים על ההתחלה.

מטרת המשחק:  
להשמיד את היחידה האויבת.

סיום המשחק: המשחק נגמר באחד משני מצבים:

1. כאשר באחד מהיחידות נהרגו כל החיילים המיוחדים

2. כאשר נגמר הזמן , המפה מתחילה להיות מופצצת , ככל שעובר יותר זמן יותר פצצות בזמן . כך שאחד מהצדדים יצתרך להפסיד על ידי פגיעה לא הוגנת של המשחק.

**ח. פרוט הדרישות מהמערכת :**

* המערכת תנהל את המשחק של השחקן ושל היריב הממוחשב .
* המערכת תבחר עבור המחשב את הצעד האופטימאלי הבא בהתאם לתנאים (חשיבה של כמה מהלכים קדימה).
* המערכת תבצע בדיקות תקינות כגון: האם אפשר לזוז לכיוון קיר, האם השחקן רואה את היריב על מנת להתחיל קרב ,האם הכדור נתקע בגבולות המפה.
* המערכת תכריז על הודעות מתאימות בכל עת כולל הכרזה על המנצח לפי תנאי סיום המשחק כפי שהוצג לעיל.
* מבנה הנתונים עליו יתבצע כל האלגוריתם והבדיקות הוא מטריצה של לוח המשחק .
* אלגוריתם האויב ימומש על ידי אלגוריתם דייקסטרה. אלגוריתם דייקסטרה פותר את בעיית מציאת המסלול הקצר ביותר מנקודה בגרף ליעד. ההיגיון באלגוריתם זה הוא שאין קצר יותר מהקצר ביותר, כלומר לא ניתן להגיע אל קדקוד בדרך קצרה יותר מאשר על ידי בחירת הדרכים הקצרות ביותר הזמינות בכל שלב.  
  האלגוריתם ישמש לשחקן המחשב (יחידת החייזרים) שיחפש את כוחות שאפשר לאסוף,יחפש את הקיר הכי קרוב להריסה (אם יש צורך) , יחפש את המיקום הכי טוב לירי על האויב ויעקוף מכשולים באמצעות אלגוריתם דייקסטרה.

**ט. מפרט טכני :**

מעבד: מעבד Pentium 4 2000 Mhz

זיכרון: RAM512.

סביבת עבודה: WINDOWS 7.

תוכנה: Java in Eclipse.

1. לוח זמנים לביצוע הפרוייקט לפי שלבים:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **11/2016** | **12/2016** | **01/2017** | **02/2017** | **03/2017** | **04/2017** | **05/2017** |
| **הכנת הצעת הפרויקט** | **X** |  |  |  |  |  |  |
| **הגשת ההצעה** | **X** |  |  |  |  |  |  |
| **תכנון** |  | **X** |  |  |  |  |  |
| **גרפיקה** |  | **X** |  |  |  |  |  |
| **כתיבה** |  |  | **X** | **X** | **X** |  |  |
| **בדיקות** |  |  |  |  | **X** | **X** |  |
| **הפעלה והגשה** |  |  |  |  |  | **X** | **X** |

# **1. מבוא:**

**1.1 מטרה:**

מטרת המשחק היא לנצח את החייזר כמה שיותר פעמים.  
השלב נגמר כאשר אחד מהצדדים מושמד על ידי הצד היריב.

**1.2 תיאור המערכת:**

המערכת מורכבת מאזור דו ממדי שגודלו דינאמי המורכב מקוביות בגודל 64X64 פיקסלים.

באיזור זה מפוזרים קירות , קופסאות והשחקנים.

במשחק יש 2 ישויות ראשיות והן השחקן והמחשב כאשר השחקן שולט על דמותו באמצעות המקלדת

והמחשב שולט על האויבים, האויב זז באמצעות אלגוריתם דייקסטרה אשר מוצא את המסלול הקצר ביותר ובכך עוזר להן להגיע לדמותו של השחקן.

**1.3 מבנה המסמך:**

הספר כולל ארבעה פרקים:

הפרק הראשון- מבוא למשחק עצמו: מטרת המשחק, תיאור כללי של המשחק, מבנה המסמך כולו ורשימת מקורות עזר בכתיבת הפרויקט.

הפרק השני- פירוט התוכנה עצמה. תיאור מפורט של המשחק, ממשק אדם-מכונה, יכולות המערכת, תיאור האלגוריתם ורשימת נתונים.

הפרק השלישי- פירוט תכנון המערכת. תיאור המודולים השונים ומבנה הנתונים.

הפרק הרביעי- הוראות הפעלה. פירוט הוראות התקנה והפעלה.

והפרק האחרון – מכיל את כל קוד המקור של הפרויקט

**1.4 רשימה ביבליוגרפית:**

1) <http://stackoverflow.com/>

2)<http://www.java-forums.org>/

3) <http://www.java2s.com/>

# **2. מפרטי תוכנה:**

יישום זה פותח בסביבת העבודה :

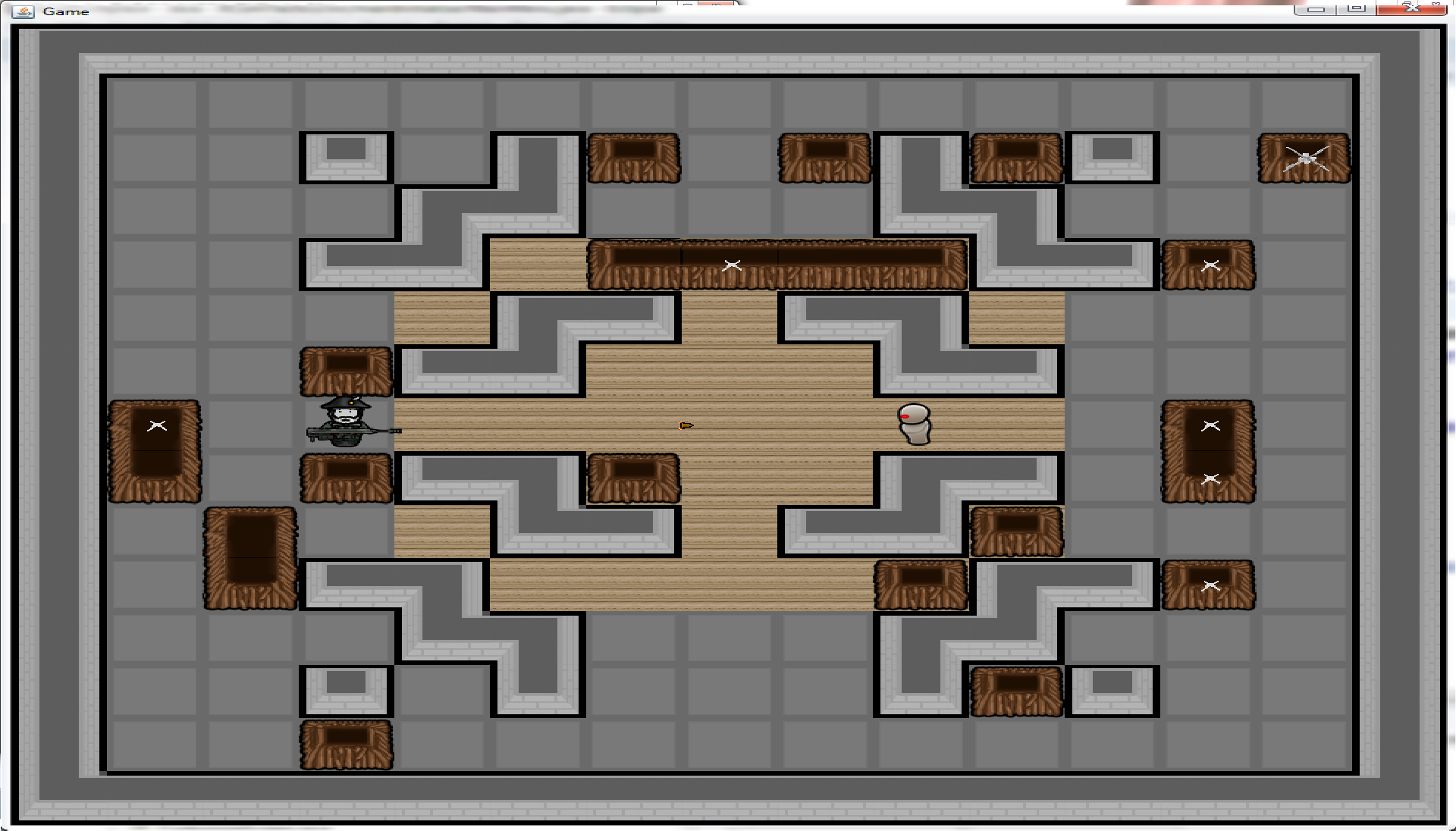


**2.1 תיאור כללי:**

המשחק מתחיל במבט מלמעלה על כל המפה

על השחקן לשרוד ולהשמיד את היריב בעזרת יריות.

השחקן זז באמצעות מקשי המקלדת wasd, ויורה כדורים באמצעות מקשי הכיוון של המקלדת.



**2.1.2 מצב של סיום משחק:**

המשחק יסתיים בהפסד כאשר מד החיים של השחקן יגיע לאפס (כל פעם שאויב פוגע בשחקן הוא מוריד לו כמות חיים מוגדרת מראש).

המשחק יסתיים בניצחון כאשר השחקן ישאר חי בלי אויבים על המפה.

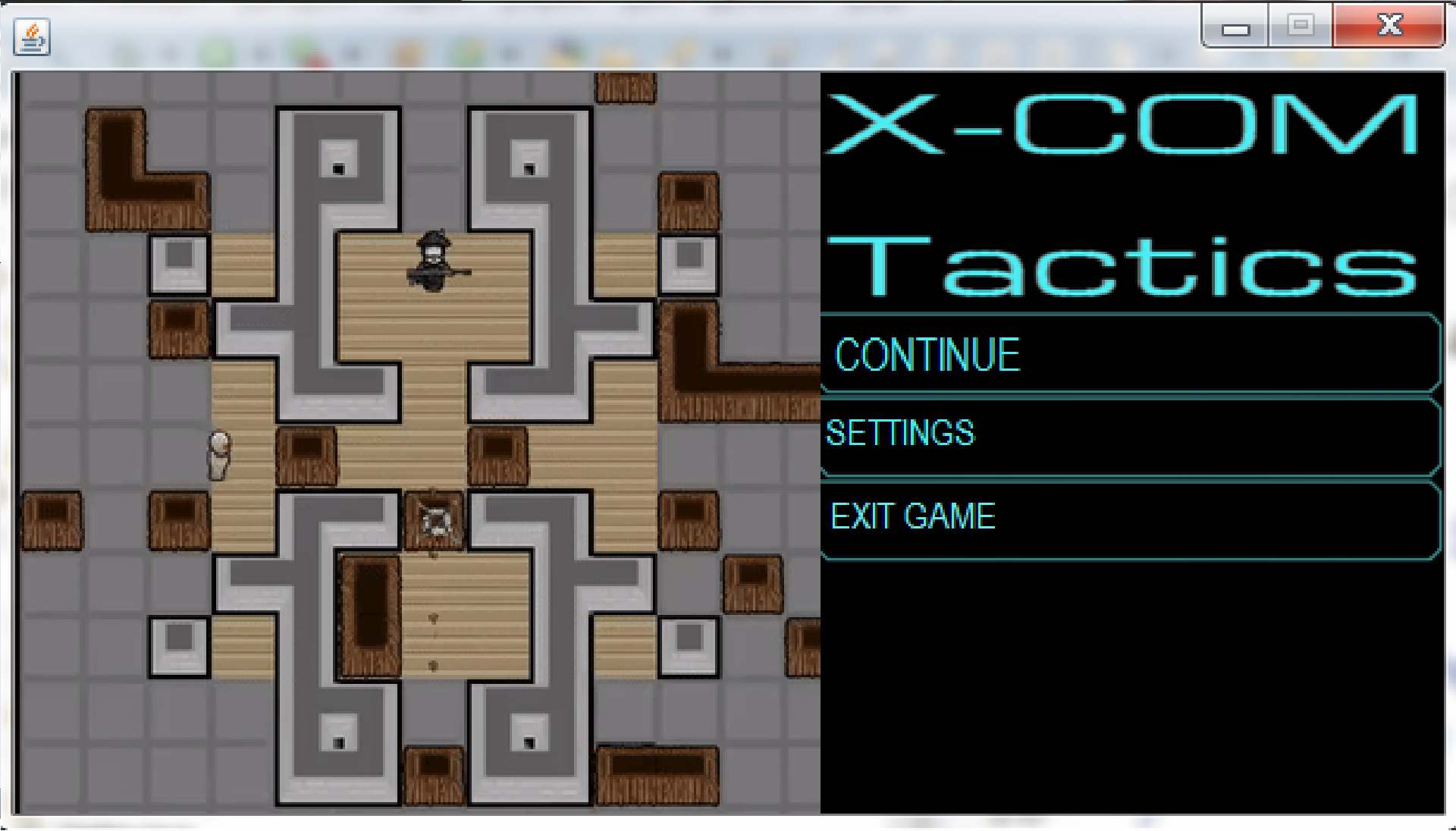
**2.1.3 אסטרטגית המשחק:**

אם האויב איטי יותר מהשחקן , יש למצוא מעגל הליכה (אפשר לחזור עליו בלי להיתקע) ולירו באויב רק כשיש בין השחקן לאויב מרחק בטוח (אסור לזוז כשיורים)

אם אין מעגל הליכה מתאים צריך ליצור אחד ( אפשר להרוס קופסאות )

אם האויב מהיר יותר צריך להתחבא כמה שיותר מאחורי קופסאות ולהישתמש בעובדה שאפשר לירו דרכהן ( אבל זה יהרוס אותן בתהליך) ולקוות שלאויב אין הרבה חיים.

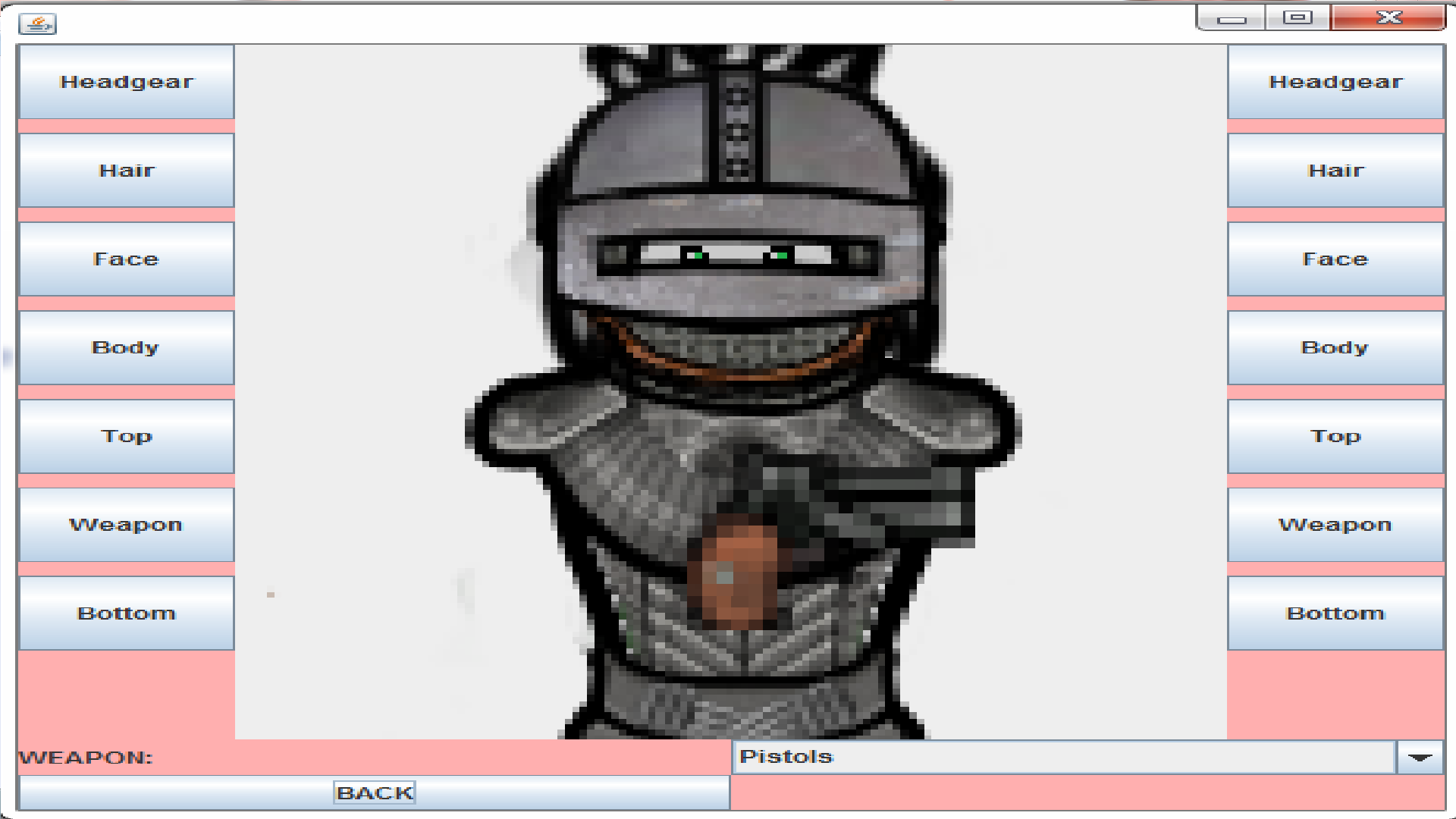
**2.2 ממשק אדם למשתמש:**



Play\Continue(1התחלת או המשך משחק.

Settings(2 פתיחת עיצוב דמות השחקן

3) Exit יציאה מהמשחק



1) שינוי כובע

2)שינוי שיער

3)שינוי פרצוף

4)שינוי גוף

5)שינוי חולצה

6)שינוי נשק

7)שינוי מכנס

8) שינוי סוג נשק

9) חזרה ל MENU

* 1. **יכולות פונקציונאליות:**

1. המערכת מאפשרת משחק של שחקן אנושי מול שחקן ממוחשב.
2. המערכת לא מאפשרת לשחקן או לאויביםלעבור דרך קירות ,שחקנים ואויבים אחרים.
3. המערכת שולטת בשימוש בכדורים\מכות משני הצדדים ויודעת לעשות פעולות בהתאם.
4. המערכת שולטת בכל חישוביי ותגובות האויב.
5. המערכת מבצעת את חישוביי התזוזה ומזיזה את האובייקטים בהתאם.
6. המערכת יוצרת שלבים חדשים בעת הפעלה של הכפתור ריסטארט וגם בתחילת הריצה של המשחק.
7. המערכת מעדכנת את מד החיים של השחקן בכל פעם שהמפלצת פוגעת בו ומעדכנת את מד החיים של האויב.
8. המערכת מודיעה לשחקן כמה חיים נשאר לו ולאויב.
   1. **תיאור אלגוריתמים :**

במשחק זה קיים האלגוריתם דייקסטרה כאלגוריתם שחקן המחשב (האויב).

**דייקסטרה:**

אלגוריתם דייקסטרה הוא אלגוריתם המוצא את המסלול הקצר ביותר מנקודה לנקודה בגרף. לאחר בחירה של קודקוד מסויים על גרף האלגוריתם יעבור על כל הקודקודים השכנים לו ויחשב את המרחקים הקצרים ביותר לכל קודקוד. חיבור המרחקים הקצרים ביותר מנקודת המקור לנקודת היעד תאפשר למצוא את המרחק המינימלי בין נקודת היעד למקור. בעזרת אלגוריתם זה יוכל המחשב לזהות את הדרך הקצרה ביותר בין מפלצת לשחקן ובכך לגרום למפלצת זו לעקוב אחר השחקן.

באלגוריתם זה משתמשים בתור קדימויות כדי למצוא את הקודקוד הבא אותו בודקים, לכן הסיבוכיות היא מספר הקשתות (כי אנו בודקים את כל הדרכים אל המטרה) ועוד מספר הקודקודים כפול לוג הקודקודים (עקב השימוש בתור קדימויות).

פסודו קוד:

Dijakstra algorithm(Graph g,Vertex source)

    create minimum priority queue Q

    foreach vertex in g do

       d[v] = infinity

       prev[v] = NIL

     d[source] = 0

    while Q is not empty

       u = the vertex in the head of Q

       for each neighbor v of u do

          temp = d[u] + length(u,v)

          if temp < d[v] do

            d[v] = temp

            prev[v] = u

     return d,prev

סיבוכיות זמן הריצה של אלגוריתם זה היא \ O(|E|+|V|log|V|)

**3. תכנון:**

**3.1 חלוקה למודלים ורשימת הנתונים:**

**1. מבנה דייקסטרה:**

מבנה המשמש למימוש אלגוריתם דייקסטרה המכיל בתוכו מבנה קודקוד וקשת למימוש גרף.

מבנה זה עוזר למחשב לחשב את הדרך הקצרה ביותר מנקודה מסויימת אל השחקן.

**2. מבנה מסך פתיחה:**

מבנה המנווט את המשתמש לריצת המשחק או לעיצוב הדמות.ללא מסך הפתיחה לא יהיה מקשר בין כל החלקים במשחק.

**3. מבנה הרנדומיזציה:**

מאפשר למשתמש לא להישתעמם מכיוון שכל מפה , תהיה שונה מהקודמת . אחראית ליצירת שני קבצים XML של רצפה וקיר.

**4. מבנה העיצוב:**

מאפשר לעצב את דמות השחקן לפי בחירתו מהמגוון הרחב שניתן לו בחלון של ההגדרות.

**5. מבנה מפה:**

מבנה המשמש לאירגון \ קריאה \ ועיצוב המפה שעליה משחק המשתמש .אחראית לציורים עדכניים על המפה בעזרת מצבים הנגרמים משאר המשחק ( יריות , הריסות)

**6. מבנה דמות על מפה:**

מבנה המכיל מידע על סוג דמות , פעולות של הדמות האפשריות חישובים עבור התנגשויות ועיצוב גרפי על המפה .

**7. מבנה כלים שימושיים:**

כלים שעושים פעולות מסוימות אך לא משויכים לשום מחלקה כמו קריאת כל הקבצים מתיקייה מסוימת עם סיומת של תמונה.

**מבנה דייקסטרה:**

מבנה המכיל את רוב המוח החשיבתי של האויב כשהמחשב משחק איתו.

**Dijektra**- מקשר את הדרכים הכי קצרות לנקודה ספציפית

|  |  |
| --- | --- |
| static void | [**computePaths**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Dijekstra.html#computePaths-engine.dijekstra.VertexInt-)(**[VertexInt](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\engine\\dijekstra\\VertexInt.html" \o "interface in engine.dijekstra)** source)  Set shortest path to each of the other player from the start **efficiency O(E+VlogV)** |
| static java.util.LinkedList<**[VertexInt](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\engine\\dijekstra\\VertexInt.html" \o "interface in engine.dijekstra)**> | [**getShortestPathTo**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Dijekstra.html#getShortestPathTo-engine.dijekstra.VertexInt-)(**[VertexInt](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\engine\\dijekstra\\VertexInt.html" \o "interface in engine.dijekstra)** target)  returns the shortest path to target target-destination **efficiency O(N)** |

**Edge**-הקשתות שמחברות בין שני ווקטורים

|  |  |
| --- | --- |
| private **[VertexInt](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\engine\\dijekstra\\VertexInt.html" \o "interface in engine.dijekstra)** | [**\_target**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Edge.html#Z:Z_target) |
| private double | [**\_weight**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Edge.html#Z:Z_weight) |

**Edge**

public Edge([VertexInt](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\engine\\dijekstra\\VertexInt.html" \o "interface in engine.dijekstra) target,

double weight)

constructor

**Parameters:**

target - - the target Vertex

weight - - the weight of the edge

**efficiency O(1)**

|  |  |
| --- | --- |
| [**VertexInt**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\VertexInt.html) | [**getTarget**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Edge.html#getTarget--)() **efficiency O(1)** |
| double | [**getWeight**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Edge.html#getWeight--)() **efficiency O(1)** |
| java.lang.String | [**toString**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Edge.html#toString--)() **efficiency O(1)** |

**Graph** - הגרף מכיל את כל הווקטורים ומסדר את מטריצת הווקטורים ככה שיהיה קשר בין כל ווקטור לשכנים שלו אם אפשר לדרוך שם

|  |  |
| --- | --- |
| private int[][] | [**\_isBlockedMat**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Graph.html#Z:Z_isBlockedMat) |
| private **[VertexInt](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\engine\\dijekstra\\VertexInt.html" \o "interface in engine.dijekstra)**[][] | [**\_matVert**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Graph.html#Z:Z_matVert) |
| private int | [**\_size**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Graph.html#Z:Z_size) |
| private int | [**\_sizeW**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Graph.html#Z:Z_sizeW) |
| private boolean | [**first**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Graph.html#first) |
| static int | [**speedToDestroyTimeDiff**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Graph.html#speedToDestroyTimeDiff) |

**Graph**

public Graph(int size,

int sizeW)

**Parameters:**

size - - the amount of rows in the graph

sizeW - - the amount of cols in the graph

**efficiency O()**

|  |  |
| --- | --- |
| void | [**addAdjaceny**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Graph.html#addAdjaceny-int-int-int-int-)(int origI, int origJ, int i, int j)  add an adjeceny from one vertex to another based on the coordinates. **efficiency O(1)** |
| void | [**buildGraph**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Graph.html#buildGraph--)()connects every vertex to all other nearby vertices **efficiency O()** |
| private boolean | [**checkIfIndexFullBlocked**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Graph.html#checkIfIndexFullBlocked-int-int-)(int i, int j)  Check if the coordinates are registered in the full wall list **efficiency O(N) where N is amount FullWalls** |
| private boolean | [**checkIfIndexHalfBlocked**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Graph.html#checkIfIndexHalfBlocked-int-int-)(int i, int j)  Check if the coordinates are registered in the half wall list **efficiency O(N) where N is amount HalfWalls** |
| [**VertexInt**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\VertexInt.html)[][] | [**getGraph**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Graph.html#getGraph--)()  get the Vertex matrix of this graph **efficiency O(1)** |

**SmartMove** -מחזיר מספר עבור המחשב שלכיוון הזה כדאי לו ללכת אם הוא מעוניין להגיע לשחקן במרחק הכי קצר

|  |  |
| --- | --- |
| private [**Graph**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Graph.html) | [**\_graph**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\SmartMove.html#Z:Z_graph) |
| private int | [**\_unitX**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\SmartMove.html#Z:Z_unitX) |
| private int | [**\_unitY**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\SmartMove.html#Z:Z_unitY) |
| private java.util.LinkedList<**[VertexInt](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\engine\\dijekstra\\VertexInt.html" \o "interface in engine.dijekstra)**> | [**ans**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\SmartMove.html#ans) |

**SmartMove**

public SmartMove()

Constructor Initiate the graph

**efficiency O()**

|  |  |
| --- | --- |
| [**Graph**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Graph.html) | [**getGraph**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\SmartMove.html#getGraph--)() **efficiency O(1)** |
| int | [**getSmartMove**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\SmartMove.html#getSmartMove-engine.Unit-int-int-)([**Unit**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Unit.html) unit, int destX, int destY)  receives the unit to start the pathing from and receives dest coordinates. **efficiency O()** |

**Vertex** - הקשת שבתוך הגרף

|  |  |
| --- | --- |
| private java.util.ArrayList<**[EdgeInt](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\engine\\dijekstra\\EdgeInt.html" \o "interface in engine.dijekstra)**> | [**\_adjacencies**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Vertex.html#Z:Z_adjacencies) |
| private int | [**\_i**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Vertex.html#Z:Z_i) |
| private int | [**\_j**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Vertex.html#Z:Z_j) |
| private double | [**\_minDistance**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Vertex.html#Z:Z_minDistance) |
| private **[VertexInt](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\engine\\dijekstra\\VertexInt.html" \o "interface in engine.dijekstra)** | [**\_previous**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Vertex.html#Z:Z_previous) |

**Vertex**

public Vertex(int i,

int j)

constructor

**Parameters:**

i - - the X coordinate of the vertex

j - - the Y coordinate of the vertex

**efficiency O(1)**

|  |  |
| --- | --- |
| void | [**clearVertex**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Vertex.html#clearVertex--)()  Nullifies the vertex for clearing previous set paths **efficiency O(1)** |
| int | [**compareTo**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Vertex.html#compareTo-engine.dijekstra.VertexInt-)(**[VertexInt](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\engine\\dijekstra\\VertexInt.html" \o "interface in engine.dijekstra)** other) **efficiency O(1)** |
| java.util.ArrayList<**[EdgeInt](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\engine\\dijekstra\\EdgeInt.html" \o "interface in engine.dijekstra)**> | [**getAdjacencies**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Vertex.html#getAdjacencies--)() **efficiency O(1)** |
| int | [**getI**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Vertex.html#getI--)() **efficiency O(1)** |
| int | [**getJ**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Vertex.html#getJ--)() **efficiency O(1)** |
| double | [**getMinDistance**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Vertex.html#getMinDistance--)() **efficiency O(1)** |
| [**VertexInt**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\VertexInt.html) | [**getPrevious**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Vertex.html#getPrevious--)() **efficiency O(1)** |
| void | [**setAdjacencies**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Vertex.html#setAdjacencies-java.util.ArrayList-)(java.util.ArrayList<**[EdgeInt](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\engine\\dijekstra\\EdgeInt.html" \o "interface in engine.dijekstra)**> adjacencies) **efficiency O(1)** |
| void | [**setMinDistance**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Vertex.html#setMinDistance-double-)(double minDistance) **efficiency O(1)** |
| void | [**setPrevious**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Vertex.html#setPrevious-engine.dijekstra.VertexInt-)(**[VertexInt](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\engine\\dijekstra\\VertexInt.html" \o "interface in engine.dijekstra)** previous) **efficiency O(1)** |
| java.lang.String | [**toString**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Vertex.html#toString--)() **efficiency O(1)** |

**מבנה מסך פתיחה:**

עוזר למשתמש להתחיל משחק וגם מחבר בין כל חלקי המשחק

**BackgroundImgPanel** -הפאנל של מסך הפתיחה במשחק מתחיל משחק \ כניסה לחלון הגדרות ויציאה נעשים כאן.

|  |  |
| --- | --- |
| private java.lang.String[] | [**\_buttonImgPath**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\BackgroundImgPanel.html#Z:Z_buttonImgPath) |
| private javax.swing.JPanel | [**\_buttonPanel**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\BackgroundImgPanel.html#Z:Z_buttonPanel) |
| private static int | [**\_currentGifIndex**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\BackgroundImgPanel.html#Z:Z_currentGifIndex) |
| private javax.swing.JButton | [**\_exitBtn**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\BackgroundImgPanel.html#Z:Z_exitBtn) |
| private **[Img](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\images\\Img.html" \o "class in images)** | [**\_gameLogo**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\BackgroundImgPanel.html#Z:Z_gameLogo) |
| private javax.swing.JLabel | [**\_gifPlayer**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\BackgroundImgPanel.html#Z:Z_gifPlayer) |
| private java.lang.String[] | [**\_gifScenePath**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\BackgroundImgPanel.html#Z:Z_gifScenePath) |
| private javax.swing.ImageIcon | [**\_imgIcon**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\BackgroundImgPanel.html#Z:Z_imgIcon) |
| private javax.swing.JButton | [**\_settingsBtn**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\BackgroundImgPanel.html#Z:Z_settingsBtn) |
| private javax.swing.JButton | [**\_startBtn**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\BackgroundImgPanel.html#Z:Z_startBtn) |
| private static **[BackgroundImgPanel](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\mainInitialize\\BackgroundImgPanel.html" \o "class in mainInitialize)** | [**singleton**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\BackgroundImgPanel.html#singleton)  the instance of the game |

**BackgroundImgPanel**

private BackgroundImgPanel(javax.swing.JLabel gifPlayer)

constructor - creates the buttons assigns button listeners and advances the gifPlayer if a new game button is clicked

**Parameters:**

main - - the frame from which the backgroundImgPanel was called

gifPlayer - - the label in which gifs will be displayed

**efficiency O(1)**

|  |  |
| --- | --- |
| static **[BackgroundImgPanel](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\mainInitialize\\BackgroundImgPanel.html" \o "class in mainInitialize)** | [**getInstance**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\BackgroundImgPanel.html#getInstance--)()  get the Singleton instance **efficiency O(1)** |
| javax.swing.JButton | [**makeButton**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\BackgroundImgPanel.html#makeButton-int-)(int i)  Make button is a function that puts an image on a button and returns the instance of it  **efficiency O(1)** |

**GameStats** - המבנה הגלובלי המכיל את כל המידע על סטטיסטיקות של שחקנים , קירות וחצאי קירות. מכיל רק רשימות רלוונטיות

|  |  |
| --- | --- |
| private static java.util.LinkedList<[**Block**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html)> | [**\_fullWalls**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\GameStats.html#Z:Z_fullWalls) |
| private static java.util.LinkedList<[**Block**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html)> | [**\_halfWalls**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\GameStats.html#Z:Z_halfWalls) |
| static int | [**\_height**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\GameStats.html#Z:Z_height) |
| private static java.util.LinkedList<java.awt.Point> | [**\_playerOccupied**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\GameStats.html#Z:Z_playerOccupied) |
| private static java.util.LinkedList<**[MovingUnit](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\MovingUnit.html" \o "class in unit)**> | [**\_players**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\GameStats.html#Z:Z_players) |
| static int | [**\_width**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\GameStats.html#Z:Z_width) |
| private static **[GameStats](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\mainInitialize\\GameStats.html" \o "class in mainInitialize)** | [**singleton**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\GameStats.html#singleton) |

**GameStats**

private GameStats()

Constructor is private because of singleton

|  |  |
| --- | --- |
| static void | [**addFullWall**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\GameStats.html#addFullWall-engine.Block-)([**Block**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html) fullWall)  add fullWall block to the fullwall list **efficiency O(1)** |
| static void | [**addHalfWall**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\GameStats.html#addHalfWall-engine.Block-)([**Block**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html) halfWall)  add halfWall block to the halfwall list **efficiency O(1)** |
| static void | [**addOccupied**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\GameStats.html#addOccupied-int-int-)(int i, int j)  add occupiedPoint block to the playerOccupied list **efficiency O(1)** |
| static void | [**addOccupied**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\GameStats.html#addOccupied-java.awt.Point-)(java.awt.Point p)  add occupiedPoint block to the playerOccupied list **efficiency O(1)** |
| static void | [**addPlayer**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\GameStats.html#addPlayer-unit.MovingUnit-)(**[MovingUnit](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\MovingUnit.html" \o "class in unit)** player)  adds a unit to the MovingUnit list **efficiency O(1)** |
| static java.util.LinkedList<[**Block**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html)> | [**getFullWalls**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\GameStats.html#getFullWalls--)()  **efficiency O(1)** |
| static java.util.LinkedList<[**Block**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html)> | [**getHalfWalls**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\GameStats.html#getHalfWalls--)()  **efficiency O(1)** |
| static **[GameStats](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\mainInitialize\\GameStats.html" \o "class in mainInitialize)** | [**getInstance**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\GameStats.html#getInstance--)()  get the Singleton instance **efficiency O(1)** |
| static java.util.LinkedList<**[MovingUnit](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\MovingUnit.html" \o "class in unit)**> | [**getPlayers**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\GameStats.html#getPlayers--)()  **efficiency O(1)** |
| static boolean | [**occupiedExists**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\GameStats.html#occupiedExists-int-int-)(int i, int j)  check if point with similar coordinates exists in playerOccupied List **efficiency O(N)** |
| static void | [**pauseGame**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\GameStats.html#pauseGame--)()  calls pauseUnit for all moving units that are registered in the Global MovingUnit list. **efficiency O(N) where N is Players Num** |
| static boolean | [**removeFullWall**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\GameStats.html#removeFullWall-engine.Block-)([**Block**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html) fullWall)  removes fullwall block from the list (uses comparison based on addresses)  **efficiency O(N)** |
| static boolean | [**removeHalfWall**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\GameStats.html#removeHalfWall-engine.Block-)([**Block**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html) halfWall)  removes halfwall block from the list (uses comparison based on addresses)  **efficiency O(N)** |
| static boolean | [**removeOccupied**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\GameStats.html#removeOccupied-int-int-)(int i, int j)  remove occupied Point from the playerOccupied list **efficiency O(N)** |
| static void | [**startGame**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\GameStats.html#startGame--)()  calls startUnit for all moving units that are registered in the Global MovingUnit list.  **efficiency O(N) where N is Players Num** |

**InstructionPanel** - המבנה שמראה את ההוראות של המשחק

|  |  |
| --- | --- |
| private javax.swing.JLabel | [**\_arrowInstruction**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\InstructionPanel.html#Z:Z_arrowInstruction) |
| private javax.swing.JButton | [**\_backBtn**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\InstructionPanel.html#Z:Z_backBtn) |
| private javax.swing.JLabel | [**\_escapeInstruction**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\InstructionPanel.html#Z:Z_escapeInstruction) |
| private javax.swing.JLabel | [**\_wasdInstruction**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\InstructionPanel.html#Z:Z_wasdInstruction) |
| private static **[InstructionPanel](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\mainInitialize\\InstructionPanel.html" \o "class in mainInitialize)** | [**singleton**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\InstructionPanel.html#singleton) |

**InstructionPanel**

private InstructionPanel()

Constructor initiates buttuns and sets listener

**efficiency O(1)**

|  |  |
| --- | --- |
| static **[InstructionPanel](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\mainInitialize\\InstructionPanel.html" \o "class in mainInitialize)** | [**getInstance**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\InstructionPanel.html#getInstance--)()  get the Singleton instance |
| static **[InstructionPanel](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\mainInitialize\\InstructionPanel.html" \o "class in mainInitialize)** | [**init**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\InstructionPanel.html#init--)()  initialize the singleton instance |

**Main** - המבנה שמריץ את כל המשחק את זה נרצה להריץ קודם

|  |  |
| --- | --- |
| static void | [**initPlayers**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\Main.html#initPlayers--)()  adds the players to the global player list and sets their places as occupied **efficiency O(1)** |
| static void | [**main**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\Main.html#main-java.lang.String:A-)(java.lang.String[] args)  the main function , runs first initiates the game **efficiency O(1)** |

**MainMenu** - הפריים של מסך הפתיחה מכיל בתוכו את הפאנל של מסך הפתיחה

|  |  |
| --- | --- |
| private **[BackgroundImgPanel](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\mainInitialize\\BackgroundImgPanel.html" \o "class in mainInitialize)** | [**\_backGroundPanel**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\MainMenu.html#Z:Z_backGroundPanel) |
| private javax.swing.JLabel | [**\_gifPlayer**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\MainMenu.html#Z:Z_gifPlayer) |
| private javax.swing.ImageIcon | [**\_imgIcon**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\MainMenu.html#Z:Z_imgIcon) |
| private static **[MainMenu](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\mainInitialize\\MainMenu.html" \o "class in mainInitialize)** | [**singleton**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\MainMenu.html#singleton) |
| private [**Unit**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Unit.html)[] | [**Units**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\MainMenu.html#Units) |

**MainMenu**

private MainMenu()

constructor is private because of singleton, puts the first gif and starts backgroundImgPanel

**efficiency O(1)**

|  |  |
| --- | --- |
| javax.swing.JLabel | [**getGifPlayer**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\MainMenu.html#getGifPlayer--)() **efficiency O(1)** |
| static **[MainMenu](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\mainInitialize\\MainMenu.html" \o "class in mainInitialize)** | [**getInstance**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\MainMenu.html#getInstance--)()  get the Singleton instance **efficiency O(1)** |
| static **[MainMenu](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\mainInitialize\\MainMenu.html" \o "class in mainInitialize)** | [**init**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\mainInitialize\MainMenu.html#init--)()  initialize the singleton instance **efficiency O(1)** |

**מבנה הרנדומיזציה:**

כל פעם יוצר מפה חדשה ראנדומלית העונה על קריטריונים.

**RandomMapCreator**

|  |  |
| --- | --- |
| private int | [**\_cols**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\RandomMapCreator.html#Z:Z_cols) |
| private int[][] | [**\_floorMap**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\RandomMapCreator.html#Z:Z_floorMap)  matrix that represents the location of diffrent floors |
| private int[] | [**\_floors**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\RandomMapCreator.html#Z:Z_floors)  floor models according to indexes |
| private int[] | [**\_keys**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\RandomMapCreator.html#Z:Z_keys)  the indexes available |
| private java.util.Random | [**\_rand**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\RandomMapCreator.html#Z:Z_rand) |
| private int | [**\_rows**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\RandomMapCreator.html#Z:Z_rows) |
| private int[][] | [**\_spawn**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\RandomMapCreator.html#Z:Z_spawn)  the spawn zone the randomization cannot override it. |
| private int[][] | [**\_wallMap**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\RandomMapCreator.html#Z:Z_wallMap)  matrix that represents the location of diffrent walls , 0 if no wall |
| private int[][] | [**\_wallModels**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\RandomMapCreator.html#Z:Z_wallModels)  wall directions used for better use of cpu , and overall better performance in order to understand - guider is the logical sequence that we can use inorder to use math to increase performance the first index represents the amount of neighbors the second index represents the index according to the logical guider |
| private int[] | [**\_walls**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\RandomMapCreator.html#Z:Z_walls)  wall models according to indexes, [0] at first means no wall ( because we read the wallmap) |

#### RandomMapCreator

public RandomMapCreator(int row,

int col)

Constructor

**Parameters:**

row - - the amount of rows in the new Generated map

col - - the amount of cols in the new Generated map

|  |  |
| --- | --- |
| void | [**createXmlFileFromMap**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\RandomMapCreator.html#createXmlFileFromMap-int:A:A-java.lang.String-)(int[][] map, java.lang.String name)  the function creates an xml file from the finalMap, with all of the elements needed **efficiency O()** |
| void | [**fillFloorMat**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\RandomMapCreator.html#fillFloorMat--)()  fill the center block with flor inside the floor map matrix **efficiency O() where N is sizeof Block** |
| void | [**fillFullWallMat**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\RandomMapCreator.html#fillFullWallMat--)()  fills symmetricaly the indestructible walls into the wallMap matrix **efficiency O() where N is sizeof HalfBorder** |
| void | [**fillHalfWallMat**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\RandomMapCreator.html#fillHalfWallMat--)()  fills the wallMap matrix with half walls randomly **efficiency O() where N is sizeof Map** |
| int | [**getModelIndex**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\RandomMapCreator.html#getModelIndex-int-int-)(int i, int j)  the function uses complex stracture of indexes I have given in order to show some logical sequence and it returns the required index of the current block **efficiency O(1)** |
| boolean | [**isFullRow**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\RandomMapCreator.html#isFullRow-int-)(int i)  boolean function that tells if the row is full with walls **efficiency O(1)** |
| boolean | [**isInBlockedZone**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\RandomMapCreator.html#isInBlockedZone-int-int-)(int i, int j)  return true if the the given indexes are occupied by value other than 0 in the spawn matrix **efficiency O(1)** |
| static int[][] | [**myCopy**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\RandomMapCreator.html#myCopy-int:A:A-int-int-)(int[][] source, int rows, int cols)  my copy , returns a new instance copy of the source . **efficiency O()** |
| int[][] | [**polishMap**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\RandomMapCreator.html#polishMap-int:A:A-)(int[][] map)  the function takes the raw drafted matrixes and converts them into walls with connection since there are 16 variations of wall directions it can become complicated if not explained properly use the template for understanding in srcImages/walls to understand the directions **efficiency O()** |
| void | [**setSpawn**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\RandomMapCreator.html#setSpawn--)()  the function sets the the corner coordinates to be key1 in the spawn matrix **efficiency O(1)** |
| int | [**sumAllWallNeighbor**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\RandomMapCreator.html#sumAllWallNeighbor-int-int-)(int i, int j)  the function sums the amount of neighbor walls in 4 direction to the current block if they are the same as current block. **efficiency O(1)** |

**מבנה עיצוב:**

מבנה המאפשר למשתמש לעצב את הדמות שלו לפי בחירתו. נפתח כאשר נלחץ כפתור ההגדרות ממסך הפתיחה

**CustomizeFrame** - מסך שבו מכיל את פאנל העיצוב של הדמות

|  |  |
| --- | --- |
| private static **[CustomizeFrame](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\customizeScreen\\CustomizeFrame.html" \o "class in unit.customizeScreen)** | [**singleton**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\CustomizeFrame.html#singleton) |

**CustomizeFrame**

private CustomizeFrame()

Constructor inits the Customize panel and shows it to the user

**efficiency O(1)**

|  |  |
| --- | --- |
| static **[CustomizeFrame](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\customizeScreen\\CustomizeFrame.html" \o "class in unit.customizeScreen)** | [**getInstance**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\CustomizeFrame.html#getInstance--)()  get the Singleton instance **efficiency O(1)** |
| static **[CustomizeFrame](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\customizeScreen\\CustomizeFrame.html" \o "class in unit.customizeScreen)** | [**init**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\CustomizeFrame.html#init--)()  initialize the singleton instance **efficiency O(1)** |

**CustomizeScreen** - פאנל המכיל בתוכו כפתורים ו פאנל של ייצוג הדמות . המשתמש לוחץ על הכפתורים עד שהדמות בייצוג תראה לו לעין.

|  |  |
| --- | --- |
| private javax.swing.JButton | [**\_backButton**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\CustomizeScreen.html#Z:Z_backButton) |
| private javax.swing.JPanel | [**\_bottomPanel**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\CustomizeScreen.html#Z:Z_bottomPanel) |
| private java.lang.String[] | [**\_nameList**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\CustomizeScreen.html#Z:Z_nameList) |
| private java.util.LinkedList<**[OptimizingButton](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\customizeScreen\\OptimizingButton.html" \o "class in unit.customizeScreen)**> | [**\_nextButtons**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\CustomizeScreen.html#Z:Z_nextButtons) |
| private javax.swing.JPanel | [**\_nextPanel**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\CustomizeScreen.html#Z:Z_nextPanel) |
| private java.util.LinkedList<**[OptimizingButton](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\customizeScreen\\OptimizingButton.html" \o "class in unit.customizeScreen)**> | [**\_prevButtons**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\CustomizeScreen.html#Z:Z_prevButtons) |
| private javax.swing.JPanel | [**\_prevPanel**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\CustomizeScreen.html#Z:Z_prevPanel) |
| private javax.swing.JComboBox<java.lang.String> | [**\_weaponSelector**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\CustomizeScreen.html#Z:Z_weaponSelector) |
| private java.lang.String[] | [**\_weaponTypes**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\CustomizeScreen.html#Z:Z_weaponTypes) |
| private static **[CustomizeScreen](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\customizeScreen\\CustomizeScreen.html" \o "class in unit.customizeScreen)** | [**singleton**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\CustomizeScreen.html#singleton) |

**CustomizeScreen**

private CustomizeScreen()

Constructor prepares all buttons and appearal sets

**efficiency O(1)**

|  |  |
| --- | --- |
| static **[CustomizeScreen](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\customizeScreen\\CustomizeScreen.html" \o "class in unit.customizeScreen)** | [**getInstance**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\CustomizeScreen.html#getInstance--)()  get the Singleton instance **efficiency O(1)** |
| java.lang.String | [**getWeaponSelected**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\CustomizeScreen.html#getWeaponSelected--)()  **efficiency O(1)** |
| static **[CustomizeScreen](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\customizeScreen\\CustomizeScreen.html" \o "class in unit.customizeScreen)** | [**init**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\CustomizeScreen.html#init--)()  initialize the singleton instance **efficiency O(1)** |
| void | [**setButtonListener**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\CustomizeScreen.html#setButtonListener-unit.customizeScreen.OptimizingButton-)(**[OptimizingButton](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\customizeScreen\\OptimizingButton.html" \o "class in unit.customizeScreen)** optBtn)  sets the button to listener for clicks **efficiency O(1)** |

**DisplayScreen** - הפאנל שמכיל את ייצוג הדמות שכרגע בעיצוב.

|  |  |
| --- | --- |
| static int | [**\_height**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\DisplayScreen.html#Z:Z_height) |
| private static **[PlayerG](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\player\\PlayerG.html" \o "class in unit.player)** | [**\_player**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\DisplayScreen.html#Z:Z_player) |
| static int | [**\_width**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\DisplayScreen.html#Z:Z_width) |
| private static **[DisplayScreen](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\customizeScreen\\DisplayScreen.html" \o "class in unit.customizeScreen)** | [**singleton**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\DisplayScreen.html#singleton) |

**DisplayScreen**

private DisplayScreen()

constructor initates the PlayerG

|  |  |
| --- | --- |
| static void | [**copyToDisplay**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\DisplayScreen.html#copyToDisplay--)()  copy the charachter from game ( if exists) to the display screen **efficiency O(1)** |
| static **[DisplayScreen](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\customizeScreen\\DisplayScreen.html" \o "class in unit.customizeScreen)** | [**getInstance**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\DisplayScreen.html#getInstance--)()  get the Singleton instance **efficiency O(1)** |
| [**PlayerG**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\PlayerG.html) | [**getPlayer**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\DisplayScreen.html#getPlayer--)()  **efficiency O(1)** |
| static **[DisplayScreen](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\customizeScreen\\DisplayScreen.html" \o "class in unit.customizeScreen)** | [**init**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\DisplayScreen.html#init--)()  initialize the singleton instance **efficiency O(1)** |
| protected void | [**paintComponent**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\DisplayScreen.html#paintComponent-java.awt.Graphics-)(java.awt.Graphics g)  The paint component calls the Paint Component of the PlayerG **efficiency O(1)** |

**OptimizingButton** - מבנה המכיל בתוכו כפתור ומקשיב ( מבנה הגרפיקה של השחקן) שמודיע על שינוי כל פעם שהוא נלחץ

|  |  |
| --- | --- |
| private javax.swing.JButton | [**\_btn**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\OptimizingButton.html#Z:Z_btn) |
| private boolean | [**\_isNext**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\OptimizingButton.html#Z:Z_isNext) |
| private **[PlayerG](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\player\\PlayerG.html" \o "class in unit.player)** | [**\_listener**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\OptimizingButton.html#Z:Z_listener) |
| private java.lang.String | [**\_name**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\OptimizingButton.html#Z:Z_name) |

#### OptimizingButton

public OptimizingButton(java.lang.String name,

boolean isNext)

Constructor

**Parameters:**

name - - the label of the button

isNext - - is it a reverse button ( next or back)

**efficiency O(1)**

|  |  |
| --- | --- |
| void | [**actionPerformed**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\OptimizingButton.html#actionPerformed-java.awt.event.ActionEvent-)(java.awt.event.ActionEvent e)  Gets called whenever the button is clicked, notifies the Listener about the click  **efficiency O(1)** |
| javax.swing.JButton | [**getBtn**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\OptimizingButton.html#getBtn--)() **efficiency O(1)** |
| java.lang.String | [**getName**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\OptimizingButton.html#getName--)() **efficiency O(1)** |
| void | [**setName**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\customizeScreen\OptimizingButton.html#setName-java.lang.String-)(java.lang.String name)  set a new Label for the button **efficiency O(1)** |

**מבנה המפה:**

המוח והגרפיקה של הלוח שעליו משחק המשתמש.

|  |  |
| --- | --- |
| private javax.swing.JButton | [**\_backToMenu**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\BrieferScreen.html#Z:Z_backToMenu) |
| private javax.swing.JLabel | [**\_briefText**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\BrieferScreen.html#Z:Z_briefText) |
| private javax.swing.JButton | [**\_newGameBtn**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\BrieferScreen.html#Z:Z_newGameBtn) |
| private static javax.swing.JDialog | [**\_parent**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\BrieferScreen.html#Z:Z_parent) |
| private javax.swing.JButton | [**\_restartBtn**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\BrieferScreen.html#Z:Z_restartBtn) |
| private static **[BrieferScreen](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\map\\BrieferScreen.html" \o "class in map)** | [**singleton**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\BrieferScreen.html#singleton) |

**BrieferScreen**

private BrieferScreen(boolean playerWon)

Constructor sets up buttons of the victory\lose screen

**Parameters:**

playerWon - - if true , Player won text will appear with new game option. else enemy won

|  |  |
| --- | --- |
| static **[BrieferScreen](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\map\\BrieferScreen.html" \o "class in map)** | [**getInstance**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\BrieferScreen.html#getInstance--)()  get the Singleton instance |
| static **[BrieferScreen](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\map\\BrieferScreen.html" \o "class in map)** | [**init**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\BrieferScreen.html#init-boolean-javax.swing.JDialog-)(boolean playerWon, javax.swing.JDialog parent)  initialize the singleton instance |
| private void | [**setUpBack**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\BrieferScreen.html#setUpBack--)()  Creates Back to menu button that closes the mapFrame and opens the menu |
| private void | [**setUpLabel**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\BrieferScreen.html#setUpLabel-boolean-)(boolean playerWon)  Creates a label with the proper img |
| private void | [**setUpNewGame**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\BrieferScreen.html#setUpNewGame--)()  Creates NewGame button that starts the game with new map |
| private void | [**setUpRestart**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\BrieferScreen.html#setUpRestart--)()  Creates Restart button that starts the game with previous map |

**MapG**  - קורא את הקבצים של המפות ויוצר BLOCK בהתאם בשביל המפה הגרפית

|  |  |
| --- | --- |
| private int | [**\_counter**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapG.html#Z:Z_counter) |
| private static [**Block**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html)[][] | [**\_map**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapG.html#Z:Z_map) |
| private int | [**\_size**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapG.html#Z:Z_size) |
| private static **[MapG](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\map\\MapG.html" \o "class in map)** | [**singleton**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapG.html#singleton) |

**MapG**

private MapG(int size,

int sizeW,

java.lang.String wallName,

java.lang.String floorName)

Constructor reads if the xml path given has childnodes and fills map with the blocks

**Parameters:**

size - - the row amount

sizeW - - the col amount

fileName - - the file path

**efficiency O()**

|  |  |
| --- | --- |
| static [**Block**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html)[][] | [**get\_map**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapG.html#get_map--)() **efficiency O(1)** |
| static **[MapG](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\map\\MapG.html" \o "class in map)** | [**getInstance**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapG.html#getInstance--)()  get the Singleton instance **efficiency O(1)** |
| static **[MapG](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\map\\MapG.html" \o "class in map)** | [**init**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapG.html#init-int-int-java.lang.String-java.lang.String-)(int size, int sizeW, java.lang.String wallName, java.lang.String floorName)  initialize the singleton instance **efficiency O(1)** |
| private void | [**readNode**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapG.html#readNode-org.w3c.dom.NodeList-org.w3c.dom.NodeList-)(org.w3c.dom.NodeList wallList, org.w3c.dom.NodeList floorList)  the function reads from the nodeList (xml) the values of the attributes and updates \_map based on it from both floorList and wallList **efficiency O()** |

**MapGFrame** - מכיל את המסך של המפה אליו מחברים את הפאנל של המפה.

|  |  |
| --- | --- |
| private static **[MapGFrame](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\map\\MapGFrame.html" \o "class in map)** | [**singleton**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapGFrame.html#singleton) |

**MapGFrame**

private MapGFrame(java.lang.String frameTitle)

Constructor

**Parameters:**

frameTitle- - the title of the frame

**efficiency O(N) where N is amount of players**

|  |  |
| --- | --- |
| void | [**EscapeButtonClicked**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapGFrame.html#EscapeButtonClicked--)()  the implemented method is called when 'escape' is clicked **efficiency O(1)** |
| static **[MapGFrame](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\map\\MapGFrame.html" \o "class in map)** | [**getInstance**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapGFrame.html#getInstance--)()  get the Singleton instance **efficiency O(1)** |
| static **[MapGFrame](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\map\\MapGFrame.html" \o "class in map)** | [**init**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapGFrame.html#init-java.lang.String-)(java.lang.String frameTitle)  initialize the singleton instance **efficiency O(1)** |

**MapGPanel** - מכיל את הפאנל של המפה , נעשים כאן חישובים על בדיקת מצבי הקירות כדי לצייר אותם בהתאם .מוסיפים לכאן כל דבר שנרצה שיופיע על המסך

|  |  |
| --- | --- |
| private int | [**\_blockSize**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapGPanel.html#Z:Z_blockSize) |
| private java.util.LinkedList<**[Img](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\images\\Img.html" \o "class in images)**> | [**\_destruction**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapGPanel.html#Z:Z_destruction) |
| private java.lang.String[] | [**\_destructionPaths**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapGPanel.html#Z:Z_destructionPaths)  All wall destructions |
| private java.util.LinkedList<**[Img](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\images\\Img.html" \o "class in images)**> | [**\_floor**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapGPanel.html#Z:Z_floor) |
| private java.lang.String[] | [**\_floorPaths**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapGPanel.html#Z:Z_floorPaths)  All Available floors |
| private int | [**\_mapHeight**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapGPanel.html#Z:Z_mapHeight) |
| private int | [**\_mapWidth**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapGPanel.html#Z:Z_mapWidth) |
| private java.util.LinkedList<**[Img](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\images\\Img.html" \o "class in images)**> | [**\_wall**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapGPanel.html#Z:Z_wall) |
| private java.lang.String[] | [**\_wallPaths**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapGPanel.html#Z:Z_wallPaths)  All Available walls |
| private boolean | [**first**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapGPanel.html#first) |
| private [**Graph**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\dijekstra\Graph.html) | [**graph**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapGPanel.html#graph) |
| java.util.LinkedList<java.awt.Point> | [**list**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapGPanel.html#list) |
| private static **[MapGPanel](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\map\\MapGPanel.html" \o "class in map)** | [**singleton**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapGPanel.html#singleton) |

**MapGPanel**

private MapGPanel(int mapHeight,

int mapWidth,

int blockSize)

constructor prepares the \_wall and \_floor and \_destruction array , also creates a new map from the two XML files

**Parameters:**

mapHeight - - the height of the map

mapWidth - - the width of the map

blockSize - - the size of a block

**efficiency O(N)**

|  |  |
| --- | --- |
| void | [**addSplitAtlasAddToList**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapGPanel.html#addSplitAtlasAddToList-java.lang.String-int-)(java.lang.String imagePath, int wallsInd)  the function receives an atlas of walls(sprites) and creates separate images and puts them into \_walls  **efficiency O(1)** |
| private int | [**getDamageIndex**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapGPanel.html#getDamageIndex-engine.Block-)([**Block**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html) block)  get the index for Destruction array based on the current health state of the block  **efficiency O(1)** |
| static **[MapGPanel](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\map\\MapGPanel.html" \o "class in map)** | [**getInstance**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapGPanel.html#getInstance--)()  get the Singleton instance  **efficiency O(1)** |
| static **[MapGPanel](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\map\\MapGPanel.html" \o "class in map)** | [**init**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapGPanel.html#init-int-int-int-)(int mapHeight, int mapWidth, int blockSize)  initialize the singleton instance  **efficiency O(1)** |
| void | [**paintComponent**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\map\MapGPanel.html#paintComponent-java.awt.Graphics-)(java.awt.Graphics g)  the paintComponent of the map  **efficiency O() where N is lengthOfMap** |
| private void | drawHUD(java.awt.Graphics g)  draws the health and scores of players and enemies at the top of the map  **efficiency O(1)** |

**Block** - מבנה המכיל בתוכו מידע שימושי עבור כל קוביה במשחק (רצפה , חצי קיר וקיר שלם)

|  |  |
| --- | --- |
| private int | [**\_bonusModel**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#Z:Z_bonusModel)  the bonusIndex according to the 4 bits before the 8 lsb bits |
| private int | [**\_currentHealth**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#Z:Z_currentHealth)  current health of a wall |
| private int | [**\_defenseVal**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#Z:Z_defenseVal)  the amount of chance to hit that is being substracted from the projectile |
| private int | [**\_floorModelIndex**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#Z:Z_floorModelIndex) |
| private int | [**\_height**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#Z:Z_height) |
| private int | [**\_imgID**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#Z:Z_imgID)  the img ID according to xml table |
| private boolean | [**\_isFull**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#Z:Z_isFull) |
| private int | [**\_maxHealth**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#Z:Z_maxHealth)  max health of a wall |
| private int | [**\_modelIndex**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#Z:Z_modelIndex)  the modelIndex according to the 4 bits before the 4lsb bits |
| private int | [**\_wallLevel**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#Z:Z_wallLevel)  diffrent levels of walls\floors |
| private int | [**\_width**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#Z:Z_width) |
| private int | [**\_x**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#Z:Z_x) |
| private int | [**\_y**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#Z:Z_y) |

**Block**

public Block(int imgID,

int x,

int y,

int width,

int height)

constructor- prepares the models and stats

**Parameters:**

imgID - - the img ID of the block according to the xml

**efficiency O(1)**

|  |  |
| --- | --- |
| boolean | [**equals**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#equals-java.lang.Object-)(java.lang.Object obj)  return true if the block contains the same wallLeven and ModelIndex in the object else false **efficiency O(1)** |
| int | [**getBonusModel**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#getBonusModel--)() **efficiency O(1)** |
| int | [**getCurrentHealth**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#getCurrentHealth--)() **efficiency O(1)** |
| int | [**getDefenseVal**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#getDefenseVal--)() **efficiency O(1)** |
| int | [**getFloorModelIndex**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#getFloorModelIndex--)() **efficiency O(1)** |
| int | [**getHeight**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#getHeight--)() **efficiency O(1)** |
| float | [**getHpDivision**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#getHpDivision--)() **efficiency O(1)** |
| int | [**getImgID**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#getImgID--)() **efficiency O(1)** |
| int | [**getMaxHealth**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#getMaxHealth--)() **efficiency O(1)** |
| int | [**getModelIndex**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#getModelIndex--)() **efficiency O(1)** |
| int | [**getProperBonusModelIndex**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#getProperBonusModelIndex-engine.Block-char-)([**Block**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html) blck, char blocker)  return the bonus model index ( direction index) for a wall based on the neighbors. **efficiency O(1)** |
| int | [**getWallLevel**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#getWallLevel--)() **efficiency O(1)** |
| int | [**getWidth**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#getWidth--)() **efficiency O(1)** |
| int | [**getX**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#getX--)() **efficiency O(1)** |
| int | [**getY**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#getY--)() **efficiency O(1)** |
| boolean | [**isFull**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#isFull--)() **efficiency O(1)** |
| void | [**setBonusModel**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#setBonusModel-int-)(int bonusModel)  set the bonusModel of the block **efficiency O(1)** |
| void | [**setCurrentHealth**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#setCurrentHealth-int-)(int currentHealth)  set the current health of the block and update accordiongly by wall from global list **efficiency O(1)** |
| void | [**setDefenseVal**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#setDefenseVal-int-)(int defenseVal)  set the defence value of the block **efficiency O(1)** |
| void | [**setFloorIndex**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#setFloorIndex-int-)(int floorModel)  set the floorModelIndex of the block and set the model index if block is floor **efficiency O(1)** |
| void | [**setHeight**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#setHeight-int-)(int height)  set the height of the block (in pixels) **efficiency O(1)** |
| void | [**setImgID**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#setImgID-int-)(int imgID)  set the ImgID of the block **efficiency O(1)** |
| void | [**setMaxHealth**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#setMaxHealth-int-)(int maxHealth)  set the max health of the block **efficiency O(1)** |
| void | [**setModelIndex**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#setModelIndex-int-)(int modelIndex)  set the modelIndex of the block **efficiency O(1)** |
| void | [**setWallLevel**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#setWallLevel-int-)(int wallLevel)  set the wall level of the block **efficiency O(1)** |
| void | [**setWidth**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#setWidth-int-)(int width)  set the width of the block (in pixels) **efficiency O(1)** |
| void | [**setX**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#setX-int-)(int x)  set the X position of the block (in position units) **efficiency O(1)** |
| void | [**setY**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#setY-int-)(int y)  set the Y position of the block (in position units) **efficiency O(1)** |
| java.lang.String | [**toString**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#toString--)() **efficiency O(1)** |
| void | [**updateNeighborsBonusModel**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html#updateNeighborsBonusModel--)()  update the neighbors of the nearby walls (used in case of removal of the block) **efficiency O(1)** |

**מבנה דמות על המפה:**

כל דבר שהוא צריך לזוז על המפה משתמש במבנה הזה אם זה שחקן או כדור היוצא מהנשק שלו.

**FireCooldown**-מבנה העשוי למנוע מדמות להיתמש בנשק שלה עד שיש אישור ממנה

|  |  |
| --- | --- |
| private int | [**\_fireCooldown**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\FireCooldown.html#Z:Z_fireCooldown) |

#### FireCooldown

public FireCooldown()**efficiency O(1)**

|  |  |
| --- | --- |
| void | [**actionPerformed**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\FireCooldown.html#actionPerformed-java.awt.event.ActionEvent-)(java.awt.event.ActionEvent e)  the action that is done whenever a timer ticks **efficiency O(1)** |
| int | [**getFireCooldown**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\FireCooldown.html#getFireCooldown--)() **efficiency O(1)** |
| void | [**setFireCooldown**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\FireCooldown.html#setFireCooldown-int-)(int fireCooldown)  set the fire cooldown for the countdown **efficiency O(1)** |

**MovingUnit** מבנה היורש ממבנה על מפה ומכיל בתוכו פונקציות של תזוזה והתקפה ( אבסטרקטית)

|  |  |
| --- | --- |
| protected **[FireCooldown](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\FireCooldown.html" \o "class in unit)** | [**\_fireListener**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\MovingUnit.html#Z:Z_fireListener) |
| protected javax.swing.Timer | [**\_fireTimer**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\MovingUnit.html#Z:Z_fireTimer) |
| protected int | [**\_i**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\MovingUnit.html#Z:Z_i)  the index of the current action ,16 is end |
| protected javax.swing.Timer | [**\_timer**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\MovingUnit.html#Z:Z_timer)  the timer that does action |

**MovingUnit**

public MovingUnit(int x,

int y,

int width,

int height) **efficiency O(1)**

|  |  |
| --- | --- |
| abstract void | [**attackDown**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\MovingUnit.html#attackDown--)()  Attack function of the movingUnit ( ranged or melee) |
| abstract void | [**attackLeft**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\MovingUnit.html#attackLeft--)()  Attack function of the movingUnit ( ranged or melee) |
| abstract void | [**attackRight**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\MovingUnit.html#attackRight--)()  Attack function of the movingUnit ( ranged or melee) |
| abstract void | [**attackUp**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\MovingUnit.html#attackUp--)()  Attack function of the movingUnit ( ranged or melee) |
| abstract int | [**getRealHeight**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\MovingUnit.html#getRealHeight--)()  get the no padding height |
| abstract int | [**getRealWidth**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\MovingUnit.html#getRealWidth--)()  get the no padding width |
| abstract int | [**getRealX**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\MovingUnit.html#getRealX--)()  get the no padding X coordinate |
| abstract int | [**getRealY**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\MovingUnit.html#getRealY--)()  get the no padding Y coordinate |
| boolean | [**isBlocked**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\MovingUnit.html#isBlocked-int-int-)(int i, int j)  Check if a certain place is blocked by other movingUnits or walls **efficiency O(1)** |
| void | [**moveDown**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\MovingUnit.html#moveDown-boolean-)(boolean isReturning)  moveDown animation uses \_i as progress does diffrent animations based on isReturning  **efficiency O(1)** |
| void | [**moveLeft**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\MovingUnit.html#moveLeft-boolean-)(boolean isReturning)  moveLeft animation uses \_i as progress does diffrent animations based on isReturning  **efficiency O(1)** |
| void | [**moveRight**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\MovingUnit.html#moveRight-boolean-)(boolean isReturning)  moveRight animation uses \_i as progress does diffrent animations based on isReturning  **efficiency O(1)** |
| void | [**moveUp**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\MovingUnit.html#moveUp-boolean-)(boolean isReturning)  moveUp animation uses \_i as progress does diffrent animations based on isReturning  **efficiency O(1)** |
| void | [**removeFromMap**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\MovingUnit.html#removeFromMap--)()  remove the moving unit from the Map and remove the global Unit List, stop all timers and nullify the graphic class  **efficiency O(1)** |

**ObjectOnMap** -מבנה של פאנל על המפה כל דבר שרוצה להופיע על המפה צריך לירוש מהמחלקה הזאתי.

|  |  |
| --- | --- |
| protected **[PaintingInterface](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\ultilityTools\\PaintingInterface.html" \o "interface in ultilityTools)** | [**\_graphicClass**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#Z:Z_graphicClass)  the graphic of the Object |
| protected int | [**\_height**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#Z:Z_height) |
| protected int | [**\_width**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#Z:Z_width) |
| protected int | [**\_x**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#Z:Z_x) |
| protected int | [**\_y**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#Z:Z_y) |
|  |  |

#### ObjectOnMap

public ObjectOnMap(int x,

int y,

int width,

int height)

Constructor

**Parameters:**

x - - the x position

y - - the y position

width - - the width of the object

height - - the height of the object

**efficiency O(1)**

|  |  |
| --- | --- |
| int | [**getObjectHeight**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#getObjectHeight--)() **efficiency O(1)** |
| int | [**getObjectWidth**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#getObjectWidth--)()  **efficiency O(1)** |
| int | [**getObjectX**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#getObjectX--)() **efficiency O(1)** |
| int | [**getObjectY**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#getObjectY--)() **efficiency O(1)** |
| boolean | [**isColliding**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#isColliding-unit.ObjectOnMap-)(**[ObjectOnMap](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\ObjectOnMap.html" \o "class in unit)** collideObj)  when the center of one panel is on the border of the other panel **efficiency O(1)** |
| boolean | [**isCollidingDown**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#isCollidingDown-unit.ObjectOnMap-)(**[ObjectOnMap](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\ObjectOnMap.html" \o "class in unit)** collideObj)  check if the current object touches the checked object from the top **efficiency O(1)** |
| boolean | [**isCollidingHorizontal**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#isCollidingHorizontal-unit.ObjectOnMap-)(**[ObjectOnMap](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\ObjectOnMap.html" \o "class in unit)** collideObj)  check if the current object touches the checked object from the left or right **efficiency O(1)** |
| boolean | [**isCollidingLeft**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#isCollidingLeft-unit.ObjectOnMap-)(**[ObjectOnMap](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\ObjectOnMap.html" \o "class in unit)** collideObj)  check if the current object touches the checked object from the right **efficiency O(1)** |
| boolean | [**isCollidingRight**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#isCollidingRight-unit.ObjectOnMap-)(**[ObjectOnMap](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\ObjectOnMap.html" \o "class in unit)** collideObj)  check if the current object touches the checked object from the left **efficiency O(1)** |
| boolean | [**isCollidingUp**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#isCollidingUp-unit.ObjectOnMap-)(**[ObjectOnMap](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\ObjectOnMap.html" \o "class in unit)** collideObj)  check if the current object touches the checked object from the bottom **efficiency O(1)** |
| boolean | [**isCollidingVertical**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#isCollidingVertical-unit.ObjectOnMap-)(**[ObjectOnMap](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\ObjectOnMap.html" \o "class in unit)** collideObj)  check if the current object touches the checked object from the top or bottom **efficiency O(1)** |
| boolean | [**isTouching**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#isTouching-unit.ObjectOnMap-)(**[ObjectOnMap](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\ObjectOnMap.html" \o "class in unit)** collideObj)  check if borders collide between an object **efficiency O(1)** |
| protected void | [**paintComponent**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#paintComponent-java.awt.Graphics-)(java.awt.Graphics g)  the painting method , if the object wants to be drawn it must implement paintingInterface **efficiency O(1)** |
| void | [**setBoundForObject**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#setBoundForObject-int-int-int-int-)(int x, int y, int width, int height)  Set bounds for the object **efficiency O(1)** |
| void | [**setObjectGraphicClass**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#setObjectGraphicClass-ultilityTools.PaintingInterface-)(**[PaintingInterface](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\ultilityTools\\PaintingInterface.html" \o "interface in ultilityTools)** graphicClass)  **efficiency O(1)** |
| void | [**setObjectHeight**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#setObjectHeight-int-)(int height)  **efficiency O(1)** |
| void | [**setObjectWidth**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#setObjectWidth-int-)(int width) **efficiency O(1)** |
| void | [**setObjectX**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#setObjectX-int-)(int x)  **efficiency O(1)** |
| void | [**setObjectY**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ObjectOnMap.html#setObjectY-int-)(int y) **efficiency O(1)** |

**Projectile** - המבנה החשיבתי של יריעה בודק התנגשויות, מחשב כמות נזק נספג והורס קירות.

|  |  |
| --- | --- |
| private double | [**\_angle**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\Projectile.html#Z:Z_angle) |
| private int | [**\_bulletDamage**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\Projectile.html#Z:Z_bulletDamage) |
| private int | [**\_bulletHp**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\Projectile.html#Z:Z_bulletHp) |
| private java.util.LinkedList<[**Block**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Block.html)> | [**\_colidedBlocks**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\Projectile.html#Z:Z_colidedBlocks) |
| private java.util.LinkedList<**[ObjectOnMap](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\ObjectOnMap.html" \o "class in unit)**> | [**\_colidedUnits**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\Projectile.html#Z:Z_colidedUnits) |
| private double | [**\_directionAngle**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\Projectile.html#Z:Z_directionAngle) |
| private **[ProjectileG](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\ProjectileG.html" \o "class in unit)** | [**\_graphics**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\Projectile.html#Z:Z_graphics) |
| private int | [**\_speed**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\Projectile.html#Z:Z_speed) |
| private javax.swing.Timer | [**\_timer**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\Projectile.html#Z:Z_timer) |
| private int | [**\_xMove**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\Projectile.html#Z:Z_xMove) |
| private int | [**\_yMove**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\Projectile.html#Z:Z_yMove) |

**Projectile**

public Projectile(int x,

int y,

int width,

int height,

int speed,

double angle,

double directionAngle)

Constructor

**Parameters:**

x - - the starting X position (in Pixels)

y - - the starting Y position (in Pixels)

width - - the width of the projectile panel

height - - the height of the projectile panel

speed - - the speed of the projectile

angle - - the angle at which to turn the projectile to

directionAngle - - the angle of cone (currently only direct projectiles)

**efficiency O(1)**

|  |  |
| --- | --- |
| void | [**actionIfCollision**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\Projectile.html#actionIfCollision--)()  do an action is struck by wall , halfwall or Moving unit  **efficiency O(N) where N is amount of blocked tiles** |
| void | [**actionPerformed**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\Projectile.html#actionPerformed-java.awt.event.ActionEvent-)(java.awt.event.ActionEvent e)  is called whenever a timer tick occours, does an action collides  with an object ,self destructs if health reaches 0  **efficiency O(N) where N is amount of blocked tiles** |
| void | [**calculateNewCordinates**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\Projectile.html#calculateNewCordinates--)()  insert the appropriate pixels moved and the height \ width of the Panel to fit the rotated projectile  **efficiency O(1)** |
| boolean | [**checkProperCollisions**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\Projectile.html#checkProperCollisions-unit.ObjectOnMap-)(**[ObjectOnMap](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\ObjectOnMap.html" \o "class in unit)** collideObj)  the function checks if collides in one of the directions  **efficiency O(1)** |
| void | [**killProjectile**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\Projectile.html#killProjectile--)()  destroy the projectile and remove from the map panel  **efficiency O(1)** |

**ProjectileG** - המבנה הגרפי של יריעה . מצייר את הכדור על המפה , ודואג להעלים אותו אם יש התרסקות בקיר מלא.

|  |  |
| --- | --- |
| private double | [**\_angle**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ProjectileG.html#Z:Z_angle) |
| private double | [**\_directionAngle**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ProjectileG.html#Z:Z_directionAngle) |
| private int | [**\_height**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ProjectileG.html#Z:Z_height) |
| private **[Img](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\images\\Img.html" \o "class in images)** | [**\_projectileImg**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ProjectileG.html#Z:Z_projectileImg) |
| private int | [**\_width**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ProjectileG.html#Z:Z_width) |

**ProjectileG**

public ProjectileG(int width,

int height,

double angle,

double directionAngle)

Constructor

**Parameters:**

width - - the width of the projectile (in pixels)

height - - the height of the projectile (in pixels)

angle - - the angle that indicates to which direction to turn the projectile

directionAngle - - the angle of the cone

**efficiency O(1)**

|  |  |
| --- | --- |
| void | [**copyProjectile**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ProjectileG.html#copyProjectile-int-int-double-double-)(int width, int height, double angle, double directionAngle)  sets the variables with the received parameters **efficiency O(1)** |
| void | [**myPaintComponent**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\ProjectileG.html#myPaintComponent-java.awt.Graphics-)(java.awt.Graphics g)  the PaintComponent of the img **efficiency O(1)** |

**מבנה שחקן על המפה:**

כל דבר הקשור ספציפית לדמות השחקן , אם זה המימוש של אנימציאת הליכה והחליפת בגדים שנבחרה על ידי המשתמש.

**ApparelSet** - מבנה הדואג לבגד מכל צד, מכיוון שצריך לצייר את הדמות ב3 מצבים (למעלה למטה וצד) צריך שיהיה איזהשהו סדר , המבנה דואג לסדר אותם בהתאם.

|  |  |
| --- | --- |
| private java.util.LinkedList<java.lang.String> | [**\_allFrontSets**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html#Z:Z_allFrontSets) |
| private java.lang.String | [**\_back**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html#Z:Z_back) |
| private int | [**\_currentSetIndex**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html#Z:Z_currentSetIndex) |
| private java.lang.String | [**\_front**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html#Z:Z_front) |
| private java.lang.String | [**\_side**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html#Z:Z_side) |

* **ApparelSet**

public ApparelSet()

constructor empty sets **efficiency O(1)**

* **ApparelSet**
* public ApparelSet(java.lang.String front,
* java.lang.String back,

java.lang.String side)

Constructor

**Parameters:**

front - - front Img Path

back - - back Img Path

side - - side Img Path

**efficiency O(1)**

|  |  |
| --- | --- |
| void | [**addFrontSet**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html#addFrontSet-java.lang.String-java.util.LinkedList-)(java.lang.String prePath, java.util.LinkedList<java.lang.String> frontPaths)  adds to allFrontSets all of the items on the list with the previous path that is received **efficiency O(N) where N sizeof frontPaths** |
| boolean | [**checkIfSamePiece**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html#checkIfSamePiece--)()  the function checks if all 3 pieces exist , and if they String Paths contain the proper endings **efficiency O(1)** |
| void | [**findCurrentSetIndex**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html#findCurrentSetIndex--)()  which front img is being used right now by comparing it to every single one and setting the currentSetIndex to it.  **efficiency O(N) where N sizeof frontPaths** |
| java.util.LinkedList<java.lang.String> | [**getAllFrontSets**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html#getAllFrontSets--)() **efficiency O(1)** |
| java.lang.String | [**getBack**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html#getBack--)() **efficiency O(1)** |
| java.lang.String | [**getFront**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html#getFront--)() **efficiency O(1)** |
| java.lang.String | [**getSide**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html#getSide--)() **efficiency O(1)** |
| void | [**putMatching**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html#putMatching-int-)(int force)  the function puts matching ImgPaths for the side that is forced.0-front,1-back,2-side **efficiency O(1)** |
| void | [**resetCurrent**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html#resetCurrent--)()  set to default (first) option for both weapons or clothing. **efficiency O(1)** |
| void | [**setBack**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html#setBack-java.lang.String-)(java.lang.String back)  set the back Image,forces matching for back **efficiency O(1)** |
| void | [**setFile**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html#setFile-java.lang.String-boolean-)(java.lang.String file, boolean isOutfit)  the function receives a file , doesnt know which ending it has. **efficiency O(1)** |
| void | [**setFront**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html#setFront-java.lang.String-boolean-)(java.lang.String front, boolean isOutfit)  set the front Image,forces matching for front **efficiency O(1)** |
| void | [**setMatchingNoDirections**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html#setMatchingNoDirections-java.lang.String-)(java.lang.String file)  exclusivly used for weapons since i have only 1 model for a weapon **efficiency O(1)** |
| void | [**setSide**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html#setSide-java.lang.String-)(java.lang.String side)  set the side Image,forces matching for side **efficiency O(1)** |
| void | [**switchTo**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html#switchTo-boolean-)(boolean isNext)  picking next item on the list or previous depends on the variable received **efficiency O(1)** |

**Outfit** - מבנה הכולל בתוכו את כל ה APPARELSETS של השחקן והוא הדבר שיטען כאשר השחקן יתחיל משחק.

|  |  |
| --- | --- |
| private **[ApparelSet](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\player\\ApparelSet.html" \o "class in unit.player)** | [**\_bottom**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#Z:Z_bottom) |
| private **[ApparelSet](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\player\\ApparelSet.html" \o "class in unit.player)** | [**\_face**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#Z:Z_face) |
| private **[FileChooser](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\ultilityTools\\FileChooser.html" \o "class in ultilityTools)** | [**\_fc**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#Z:Z_fc) |
| private **[ApparelSet](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\player\\ApparelSet.html" \o "class in unit.player)** | [**\_figure**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#Z:Z_figure) |
| private **[ApparelSet](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\player\\ApparelSet.html" \o "class in unit.player)** | [**\_hair**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#Z:Z_hair) |
| private **[ApparelSet](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\player\\ApparelSet.html" \o "class in unit.player)** | [**\_headgear**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#Z:Z_headgear) |
| private **[ApparelSet](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\player\\ApparelSet.html" \o "class in unit.player)** | [**\_top**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#Z:Z_top) |
| private **[ApparelSet](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\player\\ApparelSet.html" \o "class in unit.player)** | [**\_weapon**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#Z:Z_weapon) |

**Outfit**

public Outfit()

Constructor prepares file selector ( will be replaced with selection panel) initialitezes all apparel sets

|  |  |
| --- | --- |
| void | [**fillAllApparels**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#fillAllApparels--)()  fills all of the lists with a selection from the Clothing\Weapons folder that all ends with front **efficiency O(N) based on amount of files** |
| [**ApparelSet**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html) | [**getBottom**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#getBottom--)() **efficiency O(1)** |
| [**ApparelSet**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html) | [**getFace**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#getFace--)() **efficiency O(1)** |
| [**FileChooser**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\ultilityTools\FileChooser.html) | [**getFc**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#getFc--)() **efficiency O(1)** |
| [**ApparelSet**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html) | [**getFigure**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#getFigure--)() **efficiency O(1)** |
| [**ApparelSet**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html) | [**getHair**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#getHair--)() **efficiency O(1)** |
| [**ApparelSet**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html) | [**getHeadgear**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#getHeadgear--)() **efficiency O(1)** |
| [**ApparelSet**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html) | [**getTop**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#getTop--)() **efficiency O(1)** |
| [**ApparelSet**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\ApparelSet.html) | [**getWeapon**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#getWeapon--)() **efficiency O(1)** |
| void | [**SelectAllDefault**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#SelectAllDefault--)()  Set default apearal **efficiency O(1)** |
| void | [**setBottomFile**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#setBottomFile-java.lang.String-)(java.lang.String bottomFile)  set a new Bottom instead and isOutfit is true since its not weapon related. **efficiency O(1)** |
| void | [**setFaceFile**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#setFaceFile-java.lang.String-)(java.lang.String faceFile)  set a new Face instead and isOutfit is true since its not weapon related. **efficiency O(1)** |
| void | [**setFigureFile**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#setFigureFile-java.lang.String-)(java.lang.String figureFile)  set a new Figure instead and isOutfit is true since its not weapon related. **efficiency O(1)** |
| void | [**setHairFile**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#setHairFile-java.lang.String-)(java.lang.String hairFile)  set a new hair instead and isOutfit is true since its not weapon related. **efficiency O(1)** |
| void | [**setHeadgearFile**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#setHeadgearFile-java.lang.String-)(java.lang.String headgearFile)  set a new headgear instead and isOutfit is true since its not weapon related. **efficiency O(1)** |
| void | [**setTopFile**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#setTopFile-java.lang.String-)(java.lang.String topFile)  set a new top instead and isOutfit is true since its not weapon related. **efficiency O(1)** |
| void | [**setWeaponFile**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html#setWeaponFile-java.lang.String-)(java.lang.String weaponFile)  set a new weapon instead and isOutfit **efficiency O(1)** |

**Player** - הפאנל של השחקן שיורש מMOVINGUNIT מכיוון שהוא דמות זזה, כאן כתוב המוח של היריות ותגובה לקלט של המשתמש

|  |  |
| --- | --- |
| private boolean | [**\_busy**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#Z:Z_busy) |
| private boolean | [**\_downBusy**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#Z:Z_downBusy) |
| private **[PlayerG](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\player\\PlayerG.html" \o "class in unit.player)** | [**\_graphics**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#Z:Z_graphics)  the graphic class of the player |
| private boolean | [**\_isBlocked**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#Z:Z_isBlocked) |
| private boolean | [**\_leftBusy**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#Z:Z_leftBusy) |
| private boolean | [**\_rightBusy**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#Z:Z_rightBusy)  if the player is busy doing one of the actions the flag will be active |
| private [**Unit**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Unit.html) | [**\_stats**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#Z:Z_stats)  the stats of the player |
| private boolean | [**\_upBusy**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#Z:Z_upBusy) |

**Player**

public Player(int x,

int y,

int width,

int height)

Constructor

**Parameters:**

x - - the graphical X position of the player

y - - the graphical Y position of the player

width - - the width of the player

height - - the height of the player

map - - the Graphical map

**efficiency O(1)**

|  |  |
| --- | --- |
| void | [**actionIfWin**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#actionIfWin--)()  check if the player won , and notify the user about it  **efficiency O(N) where N is enemies amount** |
| void | [**actionPerformed**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#actionPerformed-java.awt.event.ActionEvent-)(java.awt.event.ActionEvent e)  The action that is done after every tick checks if the player won , checks if he is busy if not checks if he shoots Else check if he moves **efficiency O(1)** |
| void | [**attackDown**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#attackDown--)()  launch a projectile to the below direction if the fireCooldown is 0 **efficiency O(1)** |
| void | [**attackLeft**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#attackLeft--)()  launch a projectile to the left direction if the fireCooldown is 0 **efficiency O(1)** |
| void | [**attackRight**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#attackRight--)()  launch a projectile to the right direction if the fireCooldown is 0 **efficiency O(1)** |
| void | [**attackUp**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#attackUp--)()  launch a projectile to the upper direction if the fireCooldown is 0 **efficiency O(1)** |
| int | [**convertBooleanToInt**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#convertBooleanToInt-boolean-)(boolean bool)  convert Boolean value to integer value **efficiency O(1)** |
| [**PlayerG**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\PlayerG.html) | [**getPlayerGraphics**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#getPlayerGraphics--)() **efficiency O(1)** |
| int | [**getRealHeight**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#getRealHeight--)()  get the no padding height **efficiency O(1)** |
| int | [**getRealWidth**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#getRealWidth--)()  get the no padding width **efficiency O(1)** |
| int | [**getRealX**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#getRealX--)()  get the no padding X coordinate **efficiency O(1)** |
| int | [**getRealY**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#getRealY--)()  get the no padding Y coordinate **efficiency O(1)** |
| [**Unit**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Unit.html) | [**getStats**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#getStats--)() **efficiency O(1)** |
| boolean | [**isWin**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#isWin--)()  check if the player is the only one left in the global unit list **efficiency O(N) where N is enemies amount** |
| void | [**pauseUnit**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#pauseUnit--)()  pauses the unit , used for pausing the game **efficiency O(1)** |
| void | [**setCustomGraphics**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#setCustomGraphics-int-int-)(int height, int width)  Sets the graphics for the player based on the DisplayScreen ( if it exists) while also matching the size and setting this panel as parent **efficiency O(1)** |
| void | [**startUnit**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Player.html#startUnit--)()  starts the unit, used for continueing the game **efficiency O(1)** |

**PlayerG** - המבנה הגרפי של השחקן . מצייר את השחקן בעזרת ה OUTFIT שלו על המפה. דואג להחליף צדדים בהתאם.

|  |  |
| --- | --- |
| private **[Img](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\images\\Img.html" \o "class in images)** | [**\_bottom**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\PlayerG.html#Z:Z_bottom) |
| private [**Outfit**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html) | [**\_customize**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\PlayerG.html#Z:Z_customize) |
| private int | [**\_direction**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\PlayerG.html#Z:Z_direction) |
| private **[Img](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\images\\Img.html" \o "class in images)** | [**\_face**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\PlayerG.html#Z:Z_face) |
| private **[Img](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\images\\Img.html" \o "class in images)** | [**\_hair**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\PlayerG.html#Z:Z_hair) |
| private **[Img](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\images\\Img.html" \o "class in images)** | [**\_headGear**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\PlayerG.html#Z:Z_headGear) |
| private javax.swing.JPanel | [**\_parent**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\PlayerG.html#Z:Z_parent)  the parent is needed inorder to switch between customize screen and game |
| private **[Img](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\images\\Img.html" \o "class in images)** | [**\_top**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\PlayerG.html#Z:Z_top) |
| private **[Img](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\images\\Img.html" \o "class in images)** | [**\_unit**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\PlayerG.html#Z:Z_unit) |
| private int | [**\_unitHeight**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\PlayerG.html#Z:Z_unitHeight) |
| private int | [**\_unitWidth**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\PlayerG.html#Z:Z_unitWidth) |
| private **[Img](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\images\\Img.html" \o "class in images)** | [**\_weapon**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\PlayerG.html#Z:Z_weapon) |
| static int | [**padding**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\PlayerG.html#padding)  padding is essential for the player if we dont want some of the models to be cut. |

* **PlayerG**
* public PlayerG(int unitHeight,

int unitWidth)

Constructor

**Parameters:**

unitHeight - - the height of the player

unitWidth - - the width of the player

**efficiency O(N) based on amount of files**

* **PlayerG**
* public PlayerG(int unitHeight,
* int unitWidth,

javax.swing.JPanel parent)

Constructor

**Parameters:**

unitHeight - - the height of the player

unitWidth - - the width of the player

parent - - the Panel parent of this graphics class

**efficiency O(N) based on amount of files**

|  |  |
| --- | --- |
| void | [**copyPlayerG**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\PlayerG.html#copyPlayerG-unit.player.PlayerG-)(**[PlayerG](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\player\\PlayerG.html" \o "class in unit.player)** playerGraphics)  Copies a new players graphics to the current players graphics **efficiency O(1)** |
| void | [**CustomizeButtonClicked**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\PlayerG.html#CustomizeButtonClicked-java.lang.String-boolean-)(java.lang.String name, boolean isNext)  whenever a button is clicked in the customize menu **efficiency O(1)** |
| [**Outfit**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\Outfit.html) | [**getCustomize**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\PlayerG.html#getCustomize--)() **efficiency O(1)** |
| int | [**getDirection**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\PlayerG.html#getDirection--)() **efficiency O(1)** |
| void | [**myPaintComponent**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\PlayerG.html#myPaintComponent-java.awt.Graphics-)(java.awt.Graphics g)  the implemented method of the paintingInterface , so the Player would appear on map **efficiency O(1)** |
| void | [**setAllSizes**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\PlayerG.html#setAllSizes-int-int-)(int unitHeight, int unitWidth)  Sets all size \ positions based on the received height and width **efficiency O(1)** |
| void | [**setImgSide**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\PlayerG.html#setImgSide-int-)(int side)  select the proper img based on the side the player is facing **efficiency O(1)** |
| void | [**setParent**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\player\PlayerG.html#setParent-javax.swing.JPanel-)(javax.swing.JPanel parent)  Set a new Parent for the graphics class **efficiency O(1)** |

**מבנה אויב על המפה:**

כל דבר הקשור ספציפית לאויב , כולל ההתקפה שלו והמראה הגרפי של האויב.

**Enemy** - הפאנל של האויב שיורש מMOVINGUNIT מכיוון שהוא דמות זזה, כאן כתוב המוח של התקיפה וגם מקבל הוראה מהמהלך החכם כדי שיוכל לזוז בהתאם.

|  |  |
| --- | --- |
| private boolean | [**\_busy**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#Z:Z_busy) |
| private int | [**\_damageToWall**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#Z:Z_damageToWall) |
| private boolean | [**\_downBusy**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#Z:Z_downBusy) |
| private **[EnemyG](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\enemy\\EnemyG.html" \o "class in unit.enemy)** | [**\_graphics**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#Z:Z_graphics)  the graphic class of the enemy |
| private [**Unit**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Unit.html) | [**\_hunted**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#Z:Z_hunted) |
| private boolean | [**\_isAttacking**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#Z:Z_isAttacking) |
| private boolean | [**\_isBlocked**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#Z:Z_isBlocked) |
| private boolean | [**\_leftBusy**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#Z:Z_leftBusy) |
| private **[SmartMove](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\engine\\dijekstra\\SmartMove.html" \o "class in engine.dijekstra)** | [**\_mover**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#Z:Z_mover) |
| private int | [**\_opt**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#Z:Z_opt) |
| private boolean | [**\_rightBusy**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#Z:Z_rightBusy)  if the enemy is busy doing one of the actions the flag will be active |
| private [**Unit**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Unit.html) | [**\_stats**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#Z:Z_stats)  the stats of the enemy |
| private boolean | [**\_upBusy**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#Z:Z_upBusy) |

**Enemy**

public Enemy(int x,

int y,

int width,

int height)

Constructor

**Parameters:**

x - - the graphical X position of the enemy

y - - the graphical Y position of the enemy

width - - the width of the enemy

height - - the height of the enemy

map - - the Graphical map

**efficiency O(1)**

|  |  |
| --- | --- |
| void | [**actionIfWin**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#actionIfWin--)()  notifies the user if the enemy has won **efficiency O(N) where N is enemies amount** |
| void | [**actionPerformed**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#actionPerformed-java.awt.event.ActionEvent-)(java.awt.event.ActionEvent e)  The action that is done after every tick checks if the enemy won , checks if he is busy if not check if he needs to attack **efficiency O(1)** |
| void | [**attackDown**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#attackDown--)()  start to attack the bottom direction **efficiency O(1)** |
| void | [**attackLeft**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#attackLeft--)()  start to attack the left direction **efficiency O(1)** |
| void | [**attackRight**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#attackRight--)()  start to attack the right direction **efficiency O(1)** |
| void | [**attackUp**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#attackUp--)()  start to attack the upper direction **efficiency O(1)** |
| int | [**getDamageToWall**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#getDamageToWall--)() **efficiency O(1)** |
| int | [**getRealHeight**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#getRealHeight--)()  get the no padding height **efficiency O(1)** |
| int | [**getRealWidth**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#getRealWidth--)()  get the no padding width **efficiency O(1)** |
| int | [**getRealX**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#getRealX--)()  get the no padding X coordinate **efficiency O(1)** |
| int | [**getRealY**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#getRealY--)()  get the no padding Y coordinate **efficiency O(1)** |
| [**Unit**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\engine\Unit.html) | [**getStats**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#getStats--)() **efficiency O(1)** |
| boolean | [**isWin**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#isWin--)()  checks if there are no players left if true the enemy has won . **efficiency O(N) where N is enemies amount** |
| void | [**pauseUnit**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#pauseUnit--)()  pauses the unit , used for pausing the game **efficiency O(1)** |
| void | [**setAttacking**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#setAttacking-boolean-)(boolean isAttacking)  set the new isAttacking flag which meansif the enemy is attacking at the moment **efficiency O(1)** |
| void | [**startUnit**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html#startUnit--)()  starts the unit , used for continueing the game **efficiency O(1)** |

**EnemyAttack** - המבנה החשיבתי של התקפה בודק התנגשויות, מחשב כמות נזק נספג והורס קירות.

|  |  |
| --- | --- |
| private int | [**\_attackDamage**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyAttack.html#Z:Z_attackDamage) |
| private double | [**\_directionAngle**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyAttack.html#Z:Z_directionAngle) |
| private **[EnemyAttackG](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\enemy\\EnemyAttackG.html" \o "class in unit.enemy)** | [**\_graphics**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyAttack.html#Z:Z_graphics) |
| private [**Enemy**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html) | [**\_parent**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyAttack.html#Z:Z_parent) |
| private javax.swing.Timer | [**\_timer**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyAttack.html#Z:Z_timer) |

#### EnemyAttack

public EnemyAttack(int x,

int y,

int width,

int height,

int attackDamage,

double directionAngle,

[Enemy](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\Enemy.html) parent)

Constructor

**Parameters:**

x - - the starting X position (in Pixels)

y - - the starting Y position (in Pixels)

width - - the width of the Enemy Attack panel (in Pixels)

height - - the height of the Enemy Attack panel (in Pixels)

attackDamage - - the attack damage of the attack

directionAngle - - the direction of the attack in degrees

parent - - the parent Panel of the attack

**efficiency O(1)**

|  |  |
| --- | --- |
| Void | [**actionIfHalfWall**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyAttack.html#actionIfHalfWall--)()  do action if collides with halfwall **efficiency O(N) where N is halfwall amount** |
| Void | [**actionIfPlayer**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyAttack.html#actionIfPlayer--)()  do action if collides with player **efficiency O(N) is Players amount** |
| void | [**actionPerformed**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyAttack.html#actionPerformed-java.awt.event.ActionEvent-)(java.awt.event.ActionEvent e)  activates whenever a timer ticks  **efficiency O(1)** |
| void | [**meleeAttackFinished**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyAttack.html#meleeAttackFinished--)()  removes the enemy attack from the map when notified about animation ending. **efficiency O(1)** |

**EnemyAttackG** - המבנה הגרפי של יריעה . מצייר אנימציה של התקפה, ודואג להודיע למוח של התקיפה שנגמרה האנימציה ואפשר לתקוף שוב.

|  |  |
| --- | --- |
| private double | [**\_angle**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyAttackG.html#Z:Z_angle) |
| private **[Img](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\images\\Img.html" \o "class in images)** | [**\_attackImg**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyAttackG.html#Z:Z_attackImg) |
| private int | [**\_height**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyAttackG.html#Z:Z_height) |
| private **[MeleeAttackInterface](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\enemy\\MeleeAttackInterface.html" \o "interface in unit.enemy)** | [**\_listener**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyAttackG.html#Z:Z_listener) |
| private java.lang.String[] | [**\_pathArr**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyAttackG.html#Z:Z_pathArr) |
| private int | [**\_turn**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyAttackG.html#Z:Z_turn) |
| private int | [**\_width**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyAttackG.html#Z:Z_width) |

#### EnemyAttackG

public EnemyAttackG(int width,

int height,

double angle,

[MeleeAttackInterface](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\MeleeAttackInterface.html) listener)

Constructor

**Parameters:**

width - - the width of the attack (in pixels)

height - - the height of the attack (in pixels)

angle - - the angle of direction of the attack

listener - - the EnemyAttack that listens for the end of animation

**efficiency O(1)**

|  |  |
| --- | --- |
| void | [**myPaintComponent**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyAttackG.html#myPaintComponent-java.awt.Graphics-)(java.awt.Graphics g)  paint component for the attackImg **efficiency O(1)** |
| void | [**nextAttackImg**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyAttackG.html#nextAttackImg--)()  get the next img panel or if finished notify the listener about it **efficiency O(1)** |

**EnemyG** - המבנה הגרפי של האויב . דואג להחליף צדדים בהתאם.

|  |  |
| --- | --- |
| private int | [**\_direction**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyG.html#Z:Z_direction) |
| private **[Img](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\images\\Img.html" \o "class in images)** | [**\_unit**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyG.html#Z:Z_unit) |
| private int | [**\_unitHeight**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyG.html#Z:Z_unitHeight) |
| private **[ApparelSet](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\unit\\player\\ApparelSet.html" \o "class in unit.player)** | [**\_unitSet**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyG.html#Z:Z_unitSet) |
| private int | [**\_unitWidth**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyG.html#Z:Z_unitWidth) |

#### EnemyG

public EnemyG(int unitHeight,

int unitWidth)

Constructor

**Parameters:**

unitHeight - - the height of the Enemy Panel

unitWidth - - the width of the Enemy Panel

**efficiency O(1)**

|  |  |
| --- | --- |
| void | [**myPaintComponent**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyG.html#myPaintComponent-java.awt.Graphics-)(java.awt.Graphics g)  the implemented method of the paintingInterface , so the Player would appear on map **efficiency O(1)** |
| void | [**setAllSizes**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyG.html#setAllSizes-int-int-)(int unitHeight, int unitWidth)  set a new Size for the enemy **efficiency O(1)** |
| void | [**setImgSide**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\unit\enemy\EnemyG.html#setImgSide-int-)(int side)  select the proper img based on the side the enemy is facing **efficiency O(1)** |

**מבנה כלים שימושיים:**

כל הכלים שלא שייכים לקטע מסוים במשחק אך נורא חשובים בשביל ביצוע של המשימה

**FileChooser** - מחלקה המפעילה סייר קבצים על מקום ספציפי (לפני העיצוב היה צריך לבחור קבצים ידנית)

|  |  |
| --- | --- |
| private java.lang.String[] | [**\_filter**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\ultilityTools\FileChooser.html#Z:Z_filter) |
| private javax.swing.JFileChooser | [**\_jfc**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\ultilityTools\FileChooser.html#Z:Z_jfc) |

**FileChooser**

public FileChooser()

Constructor initiates the JFileChooser

**efficiency O(1)**

|  |  |
| --- | --- |
| java.io.File | [**choosePngFile**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\ultilityTools\FileChooser.html#choosePngFile-java.lang.String-)(java.lang.String startingFolderPath)  choose png file from the given startingFolderPath **efficiency O(N) based on files amount** |

**FileFinder** - מחלקה המחפשת קבצים בתיקייה העונים על תנאים מסוימים.

**FileFinder**

public FileFinder()

|  |  |
| --- | --- |
| static java.util.LinkedList<java.lang.String> | [**getAllFiles**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\ultilityTools\FileFinder.html#getAllFiles-java.lang.String-java.lang.String-)(java.lang.String folderPath, java.lang.String \_endingPart)  picks all files in the given folder where the name ends with a specific part  **efficiency O(N) based on files amount** |

**ImageTools** -מחלקה שימושית העושה מאניפולציה לתמונות.

#### ImageTools

public ImageTools()

|  |  |
| --- | --- |
| static java.awt.image.BufferedImage | [**createImageFromPanel**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\ultilityTools\ImageTools.html#createImageFromPanel-javax.swing.JPanel-)(javax.swing.JPanel panel)  the function draws a panel into Buffered Image which can be used again instead of drawing the components of panel again **efficiency O(1)** |
| static java.awt.image.BufferedImage | [**crop**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\ultilityTools\ImageTools.html#crop-images.Img-int-int-int-int-)(**[Img](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\images\\Img.html" \o "class in images)** imgToCrop, int cropX, int cropY, int cropHeight, int cropWidth)  Crops an Img by the new given coordinates **efficiency O(1)** |
| static java.awt.Image | [**rotate**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\ultilityTools\ImageTools.html#rotate-images.Img-double-)(**[Img](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\images\\Img.html" \o "class in images)** img, double angle)  Rotates an image. **efficiency O(1)** |
| static java.awt.image.BufferedImage | [**toBufferedImage**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\ultilityTools\ImageTools.html#toBufferedImage-images.Img-)(**[Img](file:///E:\\Programming%20Stuff\\JavaWorkplace\\Project\\XComTactics\\doc\\images\\Img.html" \o "class in images)** img)  The function converts img to buffered image since you cant always convert image to BufferedImage you will have to draw a copy of it on the new buffered Image. **efficiency O(1)** |

**Img** -מחלקת התמונות הנוחה יותר לשימוש , מכילה בתוכה דרכים שונות לציור.

|  |  |
| --- | --- |
| private java.awt.Image | [**\_image**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html#Z:Z_image) |
| private int | [**height**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html#height) |
| private int | [**width**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html#width) |
| private int | [**x**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html#x) |
| private int | [**y**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html#y) |

* **Img**

public Img() **efficiency O(1)**

* **Img**
* public Img(java.lang.String path,
* int x,
* int y,
* int width,

int height)

Constructor

**Parameters:**

path - - the path string of this Img

x - - the X position of the img

y - - the Y position of the img;

width - - the width of the img;

height - - the height of the img;

**efficiency O(1)**

|  |  |
| --- | --- |
| [**Img**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html) | [**clone**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html#clone--)()  clone is a function that returns a copy of the current Img without damaging the original **efficiency O(1)** |
| void | [**drawImg**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html#drawImg-java.awt.Graphics-)(java.awt.Graphics g)  Draws image onto graphics , using Graphics2d **efficiency O(1)** |
| void | [**drawImgHorizontally**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html#drawImgHorizontally-java.awt.Graphics-)(java.awt.Graphics g)  Draws image onto the graphics but in a horizontal position **efficiency O(1)** |
| void | [**drawImgRotate**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html#drawImgRotate-java.awt.Graphics-double-)(java.awt.Graphics g, double angle)  Draws image onto the graphics but in an angle rotation clockwise **efficiency O(1)** |
| void | [**drawImgVertically**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html#drawImgVertically-java.awt.Graphics-)(java.awt.Graphics g)  Draws image onto the graphics but in a vertical position **efficiency O(1)** |
| void | [**drawPartImage**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html#drawPartImage-java.awt.Graphics-int-int-int-int-)(java.awt.Graphics g, int newX, int newY, int newHeight, int newWidth)  Draws image onto the graphics but only a cropped part of it **efficiency O(1)** |
| void | [**drawPartImageVeritcally**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html#drawPartImageVeritcally-java.awt.Graphics-int-int-int-int-)(java.awt.Graphics g, int newX, int newY, int newHeight, int newWidth)  Draws image onto the graphics but only a cropped part of it and draw it vertically **efficiency O(1)** |
| int | [**getHeight**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html#getHeight--)() **efficiency O(1)** |
| java.awt.Image | [**getImage**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html#getImage--)() **efficiency O(1)** |
| int | [**getWidth**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html#getWidth--)() **efficiency O(1)** |
| int | [**getX**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html#getX--)() **efficiency O(1)** |
| int | [**getY**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html#getY--)() **efficiency O(1)** |
| void | [**setHeight**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html#setHeight-int-)(int height)  set a new Height for the Img **efficiency O(1)** |
| void | [**setImage**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html#setImage-java.awt.Image-)(java.awt.Image image)  set a new Image of the Img **efficiency O(1)** |
| void | [**setImage**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html#setImage-java.lang.String-)(java.lang.String path)  set a new Image of the Img **efficiency O(1)** |
| void | [**setImgCords**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html#setImgCords-int-int-)(int x, int y)  Sets image cordinates **efficiency O(1)** |
| void | [**setImgSize**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html#setImgSize-int-int-)(int width, int height)  **efficiency O(1)** |
| void | [**setWidth**](file:///E:\Programming%20Stuff\JavaWorkplace\Project\XComTactics\doc\images\Img.html#setWidth-int-)(int width)  set a new Width for the Img **efficiency O(1)** |

**3.2 מבנה הנתונים :**

מבנה הנתונים בו השתמשתי הוא גרף בעל קשרים בין הקודקודים. המפה מחולקת לבלוקים (ריבועים) ולכן כל בלוק הוא בעצם קודקוד. החיבור בין כל קודקוד יכול להיות חיבור למעלה,למטה,ימינה ושמאלה. הקשתות נעשות אם בכלל אפשרי מעבר כזה והן מקבלות משקל בהתאם להאם זה קיר נשבר או רצפה. על גרף זה רץ אלגוריתם דייקסטרה כדי למצוא את המסלול הקצר ביותר אל הקודקוד בו נמצא השחקן.

נוסף על כך השתמשתי ברשימות מקושרות ומערכים חד ממדיים ודו ממדים, אשר שימשו אותי להחזקת נתונים, למשל בכל שלב יש רשימת דמויות במשחק, כדי לבדוק האם שחקן ניצח עברתי על המערך ובדקתי האם נשאר בתוכו לפחות דמות אנושית אחת וגם אין יותר אויבים.

שמירת משתנים אלה במבנה אחיד כמו רשימה עזר לי לעקוב אחר המשתנים.

עוד דוגמאות:

רשימת מאזינים, רשימת אויבים, רשימת קירות חסימה וכ'ו.

**4. מדריך למשתמש**

**4.1 הוראות התקנה:**

יש להעתיק את התיקייה XCOM TACTICS למקום כלשהו במחשב.

**4.2 הוראות הפעלה:**

יש להיכנס לתיקייהXCom-Tactics וללחוץ דאבל קליק על קובץ המשחק Main.jar.

**4.3 הודעות מערכת:**

המערכת מודיעה דברים למשתמש בצורת תמונות על פאנל:

1. הודעה על סיום המשחק , השחקן נהרג ולכן הפסיד
2. הודעה על סיום המשחק , השחקן הרג את כל האויבים ולכן ניצח

**5. נספח – קוד מקור**

**Engine.Block**

package engine;

import mainInitialize.GameStats;

import map.MapG;

import map.MapGFrame;

import map.MapGPanel;

/\*\*

\* Block is a class that represents the current stats of the block and ImgID mapG matrix is filled with these

\*

\* @author Peter

\*/

public class Block {

/\*\*

\* the amount of chance to hit that is being substracted from the projectile

\*/

private int \_defenseVal;

/\*\*

\* max health of a wall

\*/

private int \_maxHealth;

/\*\*

\* current health of a wall

\*/

private int \_currentHealth;

/\*\*

\* diffrent levels of walls\floors

\*/

private int \_wallLevel;// 0 means floor, 1 means wall

/\*\*

\* the img ID according to xml table

\*/

private int \_imgID;

/\*\*

\* the modelIndex according to the 4 bits before the 4lsb bits

\*/

private int \_modelIndex;

/\*\*

\* the bonusIndex according to the 4 bits before the 8 lsb bits

\*/

private int \_bonusModel;

private int \_x, \_y, \_height, \_width;

private int \_floorModelIndex;

private boolean \_isFull = false;

/\*\*

\* constructor- prepares the models and stats

\*

\* @param imgID

\* - the img ID of the block according to the xml

\*/

public Block(int imgID, int x, int y, int width, int height) {

\_x = x;

\_y = y;

\_width = width;

\_height = height;

\_imgID = imgID;

\_wallLevel = (\_imgID & 0b0011);

\_modelIndex = (\_imgID >> 4) & 0b1111;

if (\_wallLevel == 0b0001) {

\_bonusModel = (\_imgID >> 8) & 0b1111;

if (\_modelIndex == 0) {

\_defenseVal = 0;

\_maxHealth = 0;

GameStats.addFullWall(this);

\_isFull = true;

} else {

GameStats.addHalfWall(this);

\_defenseVal = 30;

\_maxHealth = 10 \* 10;

}

}

\_currentHealth = \_maxHealth;

}

/\*

\* public Block(int imgID) { this(imgID,0,0,0,0); }

\*/

/\*\*

\* @return true if the wall is full else false.

\*/

public boolean isFull() {

return \_isFull;

}

/\*\*

\* @return a string containing the coordinates of the block

\*/

public String toString() {

String st = new String();

st = \_x + " " + \_y + " " + \_width + " " + \_height;

return st;

}

/\*\*

\* @return the defense value of the block

\*/

public int getDefenseVal() {

return \_defenseVal;

}

/\*\*

\* set the defence value of the block

\*

\* @param defenseVal

\* -the new defence value of the block

\*/

public void setDefenseVal(int defenseVal) {

\_defenseVal = defenseVal;

}

/\*\*

\* @return the max health of the block

\*/

public int getMaxHealth() {

return \_maxHealth;

}

/\*\*

\* set the max health of the block

\*

\* @param maxHealth

\* - the new max health of the block

\*/

public void setMaxHealth(int maxHealth) {

\_maxHealth = maxHealth;

}

/\*\*

\* @return the current health of the block

\*/

public int getCurrentHealth() {

return \_currentHealth;

}

/\*\*

\* set the floorModelIndex of the block and set the model index if block is floor

\*

\* @param floorModel

\* - the new floorModel of the block

\*/

public void setFloorIndex(int floorModel) {

\_floorModelIndex = floorModel;

if (\_wallLevel == 0)

\_modelIndex = floorModel;

}

/\*\*

\* @return the floor model index

\*/

public int getFloorModelIndex() {

return \_floorModelIndex;

}

/\*\*

\* set the current health of the block and update accordiongly by wall from global list

\*

\* @param currentHealth

\* - the new current health of the block

\*/

public void setCurrentHealth(int currentHealth) {

\_currentHealth = currentHealth;

if (\_currentHealth <= 0) {

updateNeighborsBonusModel();

\_defenseVal = 0;

\_wallLevel = 0;

\_modelIndex = \_floorModelIndex;

\_bonusModel = 0;

GameStats.removeHalfWall(this);

MapGPanel.getInstance().repaint();

// need to add update neighbor function

}

}

/\*\*

\* @return the division float between currentHealth to maxHealth

\*/

public float getHpDivision() {

return (\_maxHealth == 0) ? 0 : \_currentHealth / (float) \_maxHealth;

}

/\*\*

\* update the neighbors of the nearby walls (used in case of removal of the block)

\*/

public void updateNeighborsBonusModel() {

// System.out.println("Up");

if (MapG.get\_map()[\_x][\_y - 1].getWallLevel() != 0)

MapG.get\_map()[\_x][\_y - 1].setBonusModel(getProperBonusModelIndex(MapG.get\_map()[\_x][\_y - 1], 'd'));

// System.out.println("Down");

if (MapG.get\_map()[\_x][\_y + 1].getWallLevel() != 0)

MapG.get\_map()[\_x][\_y + 1].setBonusModel(getProperBonusModelIndex(MapG.get\_map()[\_x][\_y + 1], 'u'));

// System.out.println("Right");

if (MapG.get\_map()[\_x + 1][\_y].getWallLevel() != 0)

MapG.get\_map()[\_x + 1][\_y].setBonusModel(getProperBonusModelIndex(MapG.get\_map()[\_x + 1][\_y], 'l'));

// System.out.println("Left");

if (MapG.get\_map()[\_x - 1][\_y].getWallLevel() != 0)

MapG.get\_map()[\_x - 1][\_y].setBonusModel(getProperBonusModelIndex(MapG.get\_map()[\_x - 1][\_y], 'r'));

}

/\*\*

\* <pre>

\* return the bonus model index ( direction index) for a wall based on the neighbors.

\* REFER TO

\* \_wallModels wall directions used for better use of cpu , and overall better performance in

\* order to understand - guider is the logical sequence that we can use inorder to use math to

\* increase performance the first index represents the amount of neighbors the second index

\* represents the index according to the logical guider

\* </pre>

\*

\* \_wall3Models=//guider the wall that is missing clockwise <br/>

\* \_wall2Models//guider will be the most clockwised way exists,clockwise<br/>

\* \_wall1Models//guider the wall that exists clockwise<br/>

\*

\* @param blck

\* - the block we want to check its neighbors

\* @param blocker

\* - the Char indicating direction of the removed block

\* @return the proper direction index for the received block

\*/

public int getProperBonusModelIndex(Block blck, char blocker) {

/\*\*

\*

\*/

int[][] \_wallModels = { { 12 }, { 8, 14, 13, 4 }, { 5, 15, 10, 0, 6, 9 }, { 7, 1, 2, 11 }, { 3 } };

/\*\* if the block is a full wall \*/

if (blck.getMaxHealth() == 0)

return blck.\_bonusModel;

int i = blck.\_x, j = blck.\_y;

int up = (i > 0 && blocker != 'u' && equals(MapG.get\_map()[i][j - 1])) ? 1 : 0;

int down = (i < MapG.get\_map().length && blocker != 'd' && equals(MapG.get\_map()[i][j + 1])) ? 1 : 0;

int right = (j < MapG.get\_map().length && blocker != 'r' && equals(MapG.get\_map()[i + 1][j])) ? 1 : 0;

int left = (j > 0 && blocker != 'l' && equals(MapG.get\_map()[i - 1][j])) ? 1 : 0;

// System.out.println("up :" + up + " down :" + down + " right :" + right + " left :" +

// left);

int sum = up + down + right + left;

/\*\* if the sum of same neighbor walls is 0 or 4 \*/

if (sum % 4 == 0)

return \_wallModels[sum][0];

else {

/\*\* if the of the same neigbor walls is 1 or 3 \*/

if (sum % 2 == 1) {

if (up == down) {

if (up == right) {

return \_wallModels[sum][3];

} else {

return \_wallModels[sum][1];

}

} else {

if (up == right)

return \_wallModels[sum][0];

else {

return \_wallModels[sum][2];

}

}

}

if (up == down) {

if (up != 0)

return \_wallModels[sum][5];

else

return \_wallModels[sum][4];

} else {

if (up == left)

if (up != 0)

return \_wallModels[sum][0];

else

return \_wallModels[sum][2];

else if (up == right)

if (up != 0)

return \_wallModels[sum][1];

else

return \_wallModels[sum][3];

}

}

return 0;

}

/\*\*

\* return true if the block contains the same wallLeven and ModelIndex in the object else false

\*/

public boolean equals(Object obj) {

if (obj instanceof Block) {

Block blck = (Block) obj;

return (blck.getWallLevel() == \_wallLevel && blck.getModelIndex() == \_modelIndex);

}

return false;

}

/\*\*

\* @return the wall level of the block while 0 indicating floor and 1 indicating wall(any wall)

\*/

public int getWallLevel() {

return \_wallLevel;

}

/\*\*

\* set the wall level of the block

\*

\* @param wallLevel

\* - the new wall level of the block

\*/

public void setWallLevel(int wallLevel) {

\_wallLevel = wallLevel;

}

/\*\*

\* @return the ImgID of the block

\*/

public int getImgID() {

return \_imgID;

}

/\*\*

\* set the ImgID of the block

\*

\* @param imgID

\* - the new ImgID of the block

\*/

public void setImgID(int imgID) {

\_imgID = imgID;

}

/\*\*

\* @return the modelIndex of the block

\*/

public int getModelIndex() {

return \_modelIndex;

}

/\*\*

\* set the modelIndex of the block

\*

\* @param modelIndex

\* - the new modelIndex of the block

\*/

public void setModelIndex(int modelIndex) {

\_modelIndex = modelIndex;

}

/\*\*

\* @return the bonusModel of the block

\*/

public int getBonusModel() {

return \_bonusModel;

}

/\*\*

\* set the bonusModel of the block

\*

\* @param bonusModel

\* - the new bonus model of the block

\*/

public void setBonusModel(int bonusModel) {

\_bonusModel = bonusModel;

}

/\*\*

\* @return the X position the block (in position units)

\*/

public int getX() {

return \_x;

}

/\*\*

\* set the X position of the block (in position units)

\*

\* @param x

\* - the new X position of the block

\*/

public void setX(int x) {

\_x = x;

}

/\*\*

\* @return the Y position the block (in position units)

\*/

public int getY() {

return \_y;

}

/\*\*

\* set the Y position of the block (in position units)

\*

\* @param y

\* - the new Y position of the block

\*/

public void setY(int y) {

\_y = y;

}

/\*\*

\* @return the height of the block (in pixels)

\*/

public int getHeight() {

return \_height;

}

/\*\*

\* set the height of the block (in pixels)

\*

\* @param height

\* - the new height of the block

\*/

public void setHeight(int height) {

\_height = height;

}

/\*\*

\* @return the width of the block (in pixels)

\*/

public int getWidth() {

return \_width;

}

/\*\*

\* set the width of the block (in pixels)

\*

\* @param width

\* - the new width of the block

\*/

public void setWidth(int width) {

\_width = width;

}

}

**engine.RandomMapCreator**

package engine;

import java.util.Random;

import java.io.File;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerException;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

/\*\*

\* RandomMapCreator is a class that represents a random matrix that represents a map

\*

\* @author Peter

\*/

public class RandomMapCreator {

/\*\*

\* the indexes available

\*/

private int \_keys[] = { 0, 1, 2 };// { 128, 3077, 3173 };

/\*\*

\* matrix that represents the location of diffrent walls , 0 if no wall

\*/

private int \_wallMap[][];

/\*\*

\* matrix that represents the location of diffrent floors

\*/

private int \_floorMap[][];

private int \_rows;

private int \_cols;

/\*\*

\* wall models according to indexes, [0] at first means no wall ( because we read the wallmap)

\*/

private int[] \_walls = { 0, 0, 16 };

/\*\*

\* floor models according to indexes

\*/

private int[] \_floors = { 0, 160 };

/\*\*

\* the spawn zone the randomization cannot override it.

\*/

private int[][] \_spawn;

/\*\*

\* <pre>

\* wall directions used for better use of cpu , and overall better performance in order to

\* understand - guider is the logical sequence that we can use inorder to use math to increase

\* performance the first index represents the amount of neighbors the second index represents

\* the index according to the logical guider

\* </pre>

\*

\* \_wall3Models=//guider the wall that is missing clockwise . <br/>

\* \_wall2Models//guider will be the most clockwised wayexists, clockwise.<br/>

\* \_wall1Models//guider the wall that exists clockwise.

\*/

private int[][] \_wallModels = { { 12 }, { 8, 14, 13, 4 }, { 5, 15, 10, 0, 6, 9 }, { 7, 1, 2, 11 }, { 3 } };

private Random \_rand = new Random();

/\*\*

\* Constructor

\*

\* @param row

\* - the amount of rows in the new Generated map

\* @param col

\* - the amount of cols in the new Generated map

\*/

public RandomMapCreator(int row, int col) {

\_wallMap = new int[row][col];

\_floorMap = new int[row][col];

\_rows = row;

\_cols = col;

// one wall bonusmodel is 3072 +5

for (int i = 0; i < \_rows; i++) {

\_wallMap[i][\_cols - 1] = \_wallMap[i][0] = \_keys[1];

}

for (int i = 0; i < \_cols; i++) {

\_wallMap[\_rows - 1][i] = \_wallMap[0][i] = \_keys[1];

}

/\*\*

\* copy the walls into the spawn matrix

\*/

\_spawn = myCopy(\_wallMap, \_rows, \_cols);

setSpawn();

fillFullWallMat();

/\*\*

\* fill a block of floor in the middle

\*/

fillFloorMat();

/\*\*

\* copy the indestructible set to the spawnArray;

\*/

\_spawn = myCopy(\_wallMap, \_rows, \_cols);

setSpawn();

/\*\*

\* create randomly ,destructible blocks

\*/

fillHalfWallMat();

\_wallMap = polishMap(\_wallMap);

createXmlFileFromMap(\_floorMap, "RandomMapFloor");

createXmlFileFromMap(\_wallMap, "RandomMapWall");

}

/\*\*

\* fill the center block with flor inside the floor map matrix

\*/

public void fillFloorMat() {

for (int i = (\_rows + 1) / 4; i < (\_rows) / 2 \* 1.5; i++) {

for (int j = (\_cols + 1) / 4; j < (\_cols) / 2 \* 1.5; j++) {

if (\_wallMap[i][j] == \_wallMap[i][j] || \_wallMap[i][j] == 1)

\_floorMap[i][j] = \_floors[1];

}

}

}

/\*\*

\* fills symmetricaly the indestructible walls into the wallMap matrix

\*/

public void fillFullWallMat() {

int maxWalls, k, result;

for (int i = (\_rows + 1) / 2; i < \_rows - 2; i++) {

maxWalls = ((\_rows + 1) \* (\_cols - 1)) / 30;

for (int j = (\_cols + 1) / 2; j < \_cols - 2; j++) {

if (maxWalls > 0) {

result = \_rand.nextInt(2);

maxWalls -= (result \* 4);

} else

result = 0;

if (!isInBlockedZone(i, j) && \_wallMap[i][\_cols - 1 - j] == \_wallMap[i][j]

&& \_wallMap[\_rows - 1 - i][j] == \_wallMap[i][j]

&& \_wallMap[\_rows - 1 - i][\_cols - 1 - j] == \_wallMap[i][j]) {

\_wallMap[i][j] = \_keys[result];

\_wallMap[i][\_cols - 1 - j] = \_keys[result];

\_wallMap[\_rows - 1 - i][j] = \_keys[result];

\_wallMap[\_rows - 1 - i][\_cols - 1 - j] = \_keys[result];

}

}

}

}

/\*\*

\* fills the wallMap matrix with half walls randomly

\*/

public void fillHalfWallMat() {

int maxWalls, k;

for (int i = 1; i < \_rows - 1; i++) {

maxWalls = \_rand.nextInt(6);

while (maxWalls-- > 0 && !isFullRow(i)) {

k = \_rand.nextInt(\_cols - 2) + 1;

if ((\_wallMap[i][k] == 0) && !isInBlockedZone(i, k))

\_wallMap[i][k] = \_keys[2];

else

maxWalls++;

}

}

}

/\*\*

\* the function sets the the corner coordinates to be key1 in the spawn matrix

\*/

public void setSpawn() {

\_spawn[1][1] = \_spawn[\_rows - 2][\_cols - 2] = \_spawn[\_rows - 2][1] = \_spawn[1][\_cols - 2] = \_keys[1];

}

/\*\*

\* my copy , returns a new instance copy of the source .

\*

\* @param source

\* - the source matrix to copy

\* @param rows

\* - the rows of the new matrix

\* @param cols

\* - the cols of the new matrix

\* @return the new instance of the matrix

\*/

public static int[][] myCopy(int[][] source, int rows, int cols) {

int temp[][] = new int[rows][cols];

for (int i = 0; i < rows; i++)

for (int j = 0; j < cols; j++)

temp[i][j] = source[i][j];

return temp;

}

/\*\*

\* return true if the the given indexes are occupied by value other than 0 in the spawn matrix

\*

\* @param i

\* - the i index

\* @param j

\* - the j index

\* @return true if the indexes contain value other than 0

\*/

public boolean isInBlockedZone(int i, int j) {

if (\_spawn[i][j] != 0)

return true;

return false;

}

/\*\*

\* boolean function that tells if the row is full with walls

\*

\* @param i

\* - the index of the row

\* @return true if the row is full , false if there is at least one space available

\*/

public boolean isFullRow(int i) {

for (int j = 0; j < \_cols; j++)

if (\_wallMap[i][j] == 0)

return false;

return true;

}

/\*\*

\* the function takes the raw drafted matrixes and converts them into walls with connection

\* since there are 16 variations of wall directions it can become complicated if not explained

\* properly use the template for understanding in srcImages/walls to understand the directions

\*/

public int[][] polishMap(int map[][]) {

int tempMap[][] = RandomMapCreator.myCopy(map, \_rows, \_cols);

for (int i = 0; i < \_rows; i++) {

for (int j = 0; j < \_cols; j++) {

/\*\* if the block is a wall \*/

if (\_wallMap[i][j] != 0) {

/\*\* if the block is in the border of the map \*/

if (i % (\_rows - 1) == 0 || j % (\_cols - 1) == 0) {

/\*\* if the block is in the corner of the map \*/

if (i % (\_rows - 1) == 0 && j % (\_cols - 1) == 0) {

/\*\* if the block is in the 0 row \*/

if (i == 0) {

/\*\* if the block is in the 0 col \*/

if (j == 0) {

tempMap[i][j] = \_walls[\_wallMap[i][j]] + 5 + 10 \* 256;

}

/\*\* if the block is not in the 0 col \*/

else {

tempMap[i][j] = \_walls[\_wallMap[i][j]] + 5 + 0 \* 256;

}

}

/\*\* if the block is not in row 0 \*/

else {

/\*\* if the block is in the 0 col \*/

if (j == 0) {

tempMap[i][j] = \_walls[\_wallMap[i][j]] + 5 + 15 \* 256;

}

/\*\* if the block is not in the 0 col \*/

else {

tempMap[i][j] = \_walls[\_wallMap[i][j]] + 5 + 5 \* 256;

}

}

}

/\*\* if the block is not corner \*/

else {

/\*\* if the block is a row wall \*/

if (i % (\_rows - 1) == 0) {

tempMap[i][j] = \_walls[\_wallMap[i][j]] + 5 + 6 \* 256;

}

/\*\* if the block is a col wall \*/

if (j % (\_cols - 1) == 0) {

tempMap[i][j] = \_walls[\_wallMap[i][j]] + 5 + 9 \* 256;

}

}

}

/\*\* if the block is not border \*/

else {

// this function is really complicated to understand at

// first

tempMap[i][j] = \_walls[\_wallMap[i][j]] + 5 + 256 \* getModelIndex(i, j);

}

}

/\*\* if the block is not a wall \*/

}

}

// pass the instance of temp into wallMap

return tempMap;

}

/\*\*

\* the function sums the amount of neighbor walls in 4 direction to the current block if they

\* are the same as current block.

\*

\* @param i

\* - the row index of the current block

\* @param j

\* - the col index of the current block

\* @return the sum of all similar neighbors

\*/

public int sumAllWallNeighbor(int i, int j) {

int sum = 0;

sum += (\_wallMap[i + 1][j] == \_wallMap[i][j] ? 1 : 0) + (\_wallMap[i - 1][j] == \_wallMap[i][j] ? 1 : 0)

+ (\_wallMap[i][j + 1] == \_wallMap[i][j] ? 1 : 0) + (\_wallMap[i][j - 1] == \_wallMap[i][j] ? 1 : 0);

return sum;

}

/\*\*

\* the function uses complex stracture of indexes I have given in order to show some logical

\* sequence and it returns the required index of the current block

\*

\* @param i

\* - the row index of the current block

\* @param j

\* - the col index of the current block

\* @return the index according to the template that suits best for the block

\*/

public int getModelIndex(int i, int j) {

// if the same block as [i][j] put 1 in variable

int up = ((\_wallMap[i - 1][j] == \_wallMap[i][j]) ? 1 : 0);

int down = ((\_wallMap[i + 1][j] == \_wallMap[i][j]) ? 1 : 0);

int right = ((\_wallMap[i][j + 1] == \_wallMap[i][j]) ? 1 : 0);

int left = ((\_wallMap[i][j - 1] == \_wallMap[i][j]) ? 1 : 0);

int sum = up + down + right + left;// sumAllWallNeighbor(i, j);

/\*\* if the sum of same neighbor walls is 0 or 4 \*/

if (sum % 4 == 0)

return \_wallModels[sum][0];

else {

/\*\* if the of the same neigbor walls is 1 or 3 \*/

if (sum % 2 == 1) {

if (up == down) {

if (up == right) {

return \_wallModels[sum][3];

} else {

return \_wallModels[sum][1];

}

} else {

if (up == right)

return \_wallModels[sum][0];

else {

return \_wallModels[sum][2];

}

}

}

if (up == down) {

if (up != 0)

return \_wallModels[sum][5];

else

return \_wallModels[sum][4];

} else {

if (up == left)

if (up != 0)

return \_wallModels[sum][0];

else

return \_wallModels[sum][2];

else if (up == right)

if (up != 0)

return \_wallModels[sum][1];

else

return \_wallModels[sum][3];

}

}

return 0;

}

/\*\*

\* the function creates an xml file from the finalMap, with all of the elements needed

\* @param map - the matrix which we want to convert to xml map

\* @param name - the name of the XML file we want to give

\*/

public void createXmlFileFromMap(int map[][], String name) {

try {

DocumentBuilderFactory docFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder docBuilder = docFactory.newDocumentBuilder();

Document doc = docBuilder.newDocument();

Element rootElement = doc.createElement("Map");

doc.appendChild(rootElement);

for (int i = 0; i < \_rows; i++) {

for (int j = 0; j < \_cols; j++) {

Element Line = doc.createElement("Line");

rootElement.appendChild(Line);

Element Area = doc.createElement("Area");

Line.appendChild(Area);

Line.setAttribute("value", Integer.toString(map[i][j]));

}

}

// write the content into xml file

TransformerFactory transformerFactory = TransformerFactory.newInstance();

Transformer transformer = transformerFactory.newTransformer();

DOMSource source = new DOMSource(doc);

File f = new File("mapTemplates\\" + name + ".xml");

/\*\* if the randomMap exists , delete it \*/

if (f.exists())

f.delete();

StreamResult result = new StreamResult(f);

transformer.transform(source, result);

System.out.println("File saved!");

} catch (ParserConfigurationException pce) {

pce.printStackTrace();

} catch (TransformerException tfe) {

tfe.printStackTrace();

}

}

}

**engine.Unit**

package engine;

import unit.MovingUnit;

import unit.WinningInterface;

/\*\*

\* Unit is a class that represents the current stats of the unit

\*

\* @author Peter

\*/

public class Unit {

private char \_unitType;

private int \_health;

private int \_startingX;

private int \_startingY;

private int \_currentX;

private int \_currentY;

private MovingUnit \_listener;

/\*\*

\* Consturctor

\*

\* @param unitType

\* - the type of unit (P/E)

\* @param startX

\* - the starting X position (in position units)

\* @param startY

\* - the starting Y position (in position units)

\* @param health

\* - the health of the unit

\* @param listener

\* - the panel of the unit (that listens)

\*/

public Unit(char unitType, int startX, int startY, int health, MovingUnit listener) {

\_listener = listener;

\_unitType = unitType;

\_health = health;

\_startingX = startX;

\_currentX = \_startingX;

\_startingY = startY;

\_currentY = \_startingY;

}

/\*\*

\* @return the type of the unit

\*/

public char getUnitType() {

return \_unitType;

}

/\*\*

\* set the type of the unit

\*

\* @param unitType

\* - the new unitType

\*/

public void setUnitType(char unitType) {

\_unitType = unitType;

}

/\*\*

\* @return the health of the unit

\*/

public int getHealth() {

return \_health;

}

/\*\*

\* set the health of the unit , if health is set to zero or below remove notify listener

\*

\* @param health

\* - the new health of the unit

\*/

public void setHealth(int health) {

if (health <= 0)

\_listener.removeFromMap();

\_health = health;

}

/\*\*

\* @return the starting X coordinate of the unit (in position units)

\*/

public int getStartingX() {

return \_startingX;

}

/\*\*

\* set the starting X coordinate of the unit (in position units)

\* @param startingX

\*/

public void setStartingX(int startingX) {

\_startingX = startingX;

}

/\*\*

\* @return the starting Y coordinate of the unit (in position units)

\*/

public int getStartingY() {

return \_startingY;

}

/\*\*

\* set the starting Y coordinate of the unit (in position units)

\* @param startingY

\*/

public void setStartingY(int startingY) {

\_startingY = startingY;

}

/\*\*

\* @return the current X coordinate of the unit (in position units)

\*/

public int getCurrentX() {

return \_currentX;

}

/\*\*

\* set the current X coordinate of the unit (in position units)

\* @param currentX

\*/

public void setCurrentX(int currentX) {

\_currentX = currentX;

}

/\*\*

\* @return the current Y coordinate of the unit (in position units)

\*/

public int getCurrentY() {

return \_currentY;

}

/\*\*

\* set the current Y coordinate of the unit (in position units)

\* @param currentY

\*/

public void setCurrentY(int currentY) {

\_currentY = currentY;

}

/\*\*

\* return a string containing the unit type

\*/

public String toString() {

return "Unit " + \_unitType;

}

}

**engine.dijekstra.Dijekstra**

package engine.dijekstra;

import java.util.\*;

import engine.dijekstra.EdgeInt;

import engine.dijekstra.VertexInt;

/\*\*

\* Class for Dijkstra

\*

\* @author Peter

\*/

public class Dijekstra {

/\*\*

\* Find the best short tracks enemy (policeman) to each of the other vertices

\*

\* @param source

\* -the vertex we want to start the procedure from

\*/

public static void computePaths(VertexInt source) {

source.setMinDistance(0.0);

PriorityQueue<VertexInt> vertexQueue = new PriorityQueue<VertexInt>();

vertexQueue.add(source);

while (!vertexQueue.isEmpty()) {

VertexInt current = vertexQueue.poll();

for (EdgeInt e : current.getAdjacencies()) {

VertexInt target = e.getTarget();

double distance = e.getWeight();

double distanceThroughU = current.getMinDistance() + distance;

if (distanceThroughU < target.getMinDistance()) {

vertexQueue.remove(target);

target.setMinDistance(distanceThroughU);

target.setPrevious(current);

vertexQueue.add(target);

}

}

}

}

/\*\*

\* returns the shortest path to target target-destination

\*

\* @param target

\* - the the shortest path to target from the previous given computePaths

\*/

public static LinkedList<VertexInt> getShortestPathTo(VertexInt target) {

LinkedList<VertexInt> path = new LinkedList<VertexInt>();

for (VertexInt vertex = target; vertex != null; vertex = vertex.getPrevious()) {

path.add(vertex);

}

Collections.reverse(path);

return path;

}

}

**engine.dijekstra.Edge**

package engine.dijekstra;

/\*\*

\* The edge represents the one side connection between two vertices

\* @author Peter

\*

\*/

public class Edge implements EdgeInt {

private final VertexInt \_target;

private final double \_weight;

/\*\*

\* constructor

\* @param target - the target Vertex

\* @param weight - the weight of the edge

\*/

public Edge(VertexInt target, double weight) {

\_target = target;

\_weight = weight;

}

/\*\*

\* @return the target Vertex of this edge

\*/

@Override

public VertexInt getTarget() {

return \_target;

}

/\*\*

\* @return the weight of this edge

\*/

@Override

public double getWeight() {

return \_weight;

}

/\*\*

\* @return the string representing the Target and weight

\*/

@Override

public String toString() {

return "T:" + \_target + " W:" + \_weight;

}

}

**engine.dijekstra.EdgeInt**

package engine.dijekstra;

public interface EdgeInt {

public VertexInt getTarget();

public double getWeight();

}

**engine.dijekstra.Graph**

package engine.dijekstra;

import java.util.ArrayList;

import unit.enemy.Enemy;

import engine.Block;

import mainInitialize.GameStats;

import map.MapG;

/\*\*

\* Graph contains all of the vertices and manipulates and connects every vertex with all other

\* neighbor vertices through using edges

\*

\* @author Peter

\*/

public class Graph implements GraphInt {

private int[][] \_isBlockedMat;

private VertexInt[][] \_matVert;

private int \_size;

private int \_sizeW;

public static int speedToDestroyTimeDiff = 5;// 120/((Enemy)(GameStats.getPlayers().getLast())).getDamageToWall();

private boolean first = true;

/\*\*

\*

\* @param size - the amount of rows in the graph

\* @param sizeW - the amount of cols in the graph

\*/

public Graph(int size, int sizeW) {

\_size = size;

\_sizeW = sizeW;

\_isBlockedMat = new int[\_size][\_sizeW];

\_matVert = new VertexInt[\_size][\_sizeW];

for (int i = 0; i < \_size; i++)

for (int j = 0; j < \_sizeW; j++) {

\_matVert[i][j] = new Vertex(i, j);

}

}

/\*\*

\* get the Vertex matrix of this graph

\*/

@Override

public VertexInt[][] getGraph() {

return \_matVert;

}

/\*\*

\* Check if the coordinates are registered in the full wall list

\* @param i - the X coordinate

\* @param j - the Y coordinate

\* @return true if it exists else false.

\*/

private boolean checkIfIndexFullBlocked(int i, int j) {

for (Block temp : GameStats.getFullWalls()) {

if (temp.getX() == i && temp.getY() == j)

return true;

}

return false;

}

/\*\*

\* Check if the coordinates are registered in the half wall list

\* @param i - the X coordinate

\* @param j - the Y coordinate

\* @return true if it exists else false.

\*/

private boolean checkIfIndexHalfBlocked(int i, int j) {

for (Block temp : GameStats.getHalfWalls()) {

if (temp.getX() == i && temp.getY() == j)

return true;

}

return false;

}

/\*\*

\* add an adjeceny from one vertex to another based on the coordinates.

\* @param origI - the X coordinate of the source

\* @param origJ - the Y coordinate of the source

\* @param i - the X coordinate of the dest

\* @param j - the Y coordinate of the dest

\*/

public void addAdjaceny(int origI, int origJ, int i, int j) {

\_matVert[origI][origJ].getAdjacencies()

.add(new Edge(\_matVert[i][j], (checkIfIndexHalfBlocked(i, j)) ? speedToDestroyTimeDiff : 1));

}

/\*\*

\* connects every vertex to all other nearby vertices

\*/

@Override

public void buildGraph() {

for (int i = 0; i < \_size; i++) {

for (int j = 0; j < \_sizeW; j++) {

if (!MapG.get\_map()[i][j].isFull()) {

\_matVert[i][j].getAdjacencies().clear();

if (!MapG.get\_map()[i - 1][j].isFull()) {

addAdjaceny(i, j, i - 1, j);

}

if (!MapG.get\_map()[i + 1][j].isFull()) {

addAdjaceny(i, j, i + 1, j);

}

if (!MapG.get\_map()[i][j - 1].isFull()) {

addAdjaceny(i, j, i, j - 1);

}

if (!MapG.get\_map()[i][j + 1].isFull()) {

addAdjaceny(i, j, i, j + 1);

}

}

}

}

}

}

**engine.dijekstra.GraphInt**

package engine.dijekstra;

public interface GraphInt {

public VertexInt[][] getGraph();

public void buildGraph();

}

**engine.dijekstra.SmartMove**

package engine.dijekstra;

public interface GraphInt {

public VertexInt[][] getGraph();

public void buildGraph();

}

**engine.dijekstra.Vertex**

package engine.dijekstra;

import java.util.ArrayList;

/\*\*

\* Vertex is a class that represents a vertex in a graph

\*

\* @author Peter

\*/

public class Vertex implements VertexInt {

private ArrayList<EdgeInt> \_adjacencies;

private double \_minDistance = Double.POSITIVE\_INFINITY;

private VertexInt \_previous;

private int \_i, \_j;

/\*\*

\* constructor

\*

\* @param i

\* - the X coordinate of the vertex

\* @param j

\* - the Y coordinate of the vertex

\*/

public Vertex(int i, int j) {

\_i = i;

\_j = j;

\_adjacencies = new ArrayList<EdgeInt>();

}

@Override

public int getI() {

return \_i;

}

@Override

public int getJ() {

return \_j;

}

@Override

public ArrayList<EdgeInt> getAdjacencies() {

return \_adjacencies;

}

@Override

public void setAdjacencies(ArrayList<EdgeInt> adjacencies) {

\_adjacencies = adjacencies;

}

@Override

public double getMinDistance() {

return \_minDistance;

}

@Override

public void setMinDistance(double minDistance) {

\_minDistance = minDistance;

}

@Override

public VertexInt getPrevious() {

return \_previous;

}

@Override

public void setPrevious(VertexInt previous) {

\_previous = previous;

}

@Override

public int compareTo(VertexInt other) {

return Double.compare(\_minDistance, other.getMinDistance());

}

/\*\*

\* Nullifies the vertex for clearing previous set paths

\*/

@Override

public void clearVertex() {

setMinDistance(Double.POSITIVE\_INFINITY);

setPrevious(null);

}

@Override

public String toString() {

return \_i + "," + \_j;

}

}

**engine.dijekstra.VertexInt**

package engine.dijekstra;

import java.util.ArrayList;

public interface VertexInt extends Comparable<VertexInt> {

public int getI();

public int getJ();

public ArrayList<EdgeInt> getAdjacencies();

public void setAdjacencies(ArrayList<EdgeInt> adjacencies);

public double getMinDistance();

public void setMinDistance(double minDistance);

public VertexInt getPrevious();

public void setPrevious(VertexInt previous);

public int compareTo(VertexInt other);

public void clearVertex();

}

**engine.input.KeyboardInput**

package engine.input;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.util.LinkedList;

import javax.swing.JFrame;

import javax.swing.JPanel;

import map.EscapeButtonInterface;

/\*\*

\* KeyboardInput is a class that is used for inputing keyboard input from the user , extends JPanel because it is an invisible Panel , and implements KeyListener because it listens to keyboard clicks.

\* <br/>

\* Since the class is a singleton , it uses getInstance to give out the static appearance. if an outer class wants to use the keyboard input it should use addListener and send itselves there.

\* @category Singleton - only contains one instance

\* @author Peter

\*

\*/

public class KeyboardInput extends JPanel implements KeyListener {

/\*\* singleton \*/

private static KeyboardInput \_singleton = new KeyboardInput();

/\*\* the list of listeners \*/

private LinkedList<Object> \_listenerList;

/\*\* the list of all keyboard keys \*/

private LinkedList<Boolean> \_keys;

/\*\*

\* the constructor creates a new Listener list and adds 256 keys according to ascii code.<br/>

\* constructor is private because of singleton.

\*/

private KeyboardInput() {

\_listenerList = new LinkedList<Object>();

\_keys = new LinkedList<Boolean>();

for (int i = 0; i < 256; i++) {

\_keys.add(false);

}

}

/\*\*

\* get the Singleton instance

\*

\* @return KeyboardInput only instance

\*/

public static KeyboardInput getInstance() {

return \_singleton;

}

/\*\*

\* the function receives an object which could be anything therefore the conversions to frame or panel ,and adds to that object a key Listener which is the current KeyboardInput

\*

\* @param listener

\* - the object that wants to listen to events

\*/

public void addListener(Object listener) {

if (listener instanceof JFrame)

((JFrame) listener).addKeyListener(this);

else if (listener instanceof JPanel)

((JPanel) listener).addKeyListener(this);

\_listenerList.addLast(listener);

}

/\*\*

\* set the value the key in the list

\*/

public void setValue(int index, boolean value) {

\_keys.set(index, value);

}

/\*\*

\* the function alert all listeners of the required interfaces

\*/

public void alertListeners() {

/\*\*

\* if we need to alert all EscapeButtonListeners check that they actually want to listen to EscapeButtons

\*/

if (\_keys.get(KeyEvent.VK\_ESCAPE))

for (int i = 0; i < \_listenerList.size(); i++)

if (\_listenerList.get(i) instanceof EscapeButtonInterface) {

((EscapeButtonInterface) \_listenerList.get(i)).EscapeButtonClicked();

\_keys.set(KeyEvent.VK\_ESCAPE, false);

}

}

/\*\*

\* activate the key accordiongly to the key pressed

\*/

@Override

public void keyPressed(KeyEvent e) {

try {

\_keys.set(e.getKeyCode(), true);

} catch (IndexOutOfBoundsException b) {

}

alertListeners();

}

/\*\*

\* disable the key accordiongly to the key released

\*/

@Override

public void keyReleased(KeyEvent arg0) {

try {

\_keys.set(arg0.getKeyCode(), false);

} catch (IndexOutOfBoundsException e) {

}

}

@Override

public void keyTyped(KeyEvent arg0) {

}

/\*\*

\* @return the list of all boolean keyboard keys

\*/

public LinkedList<Boolean> get\_keys() {

return \_keys;

}

}

**images.Img**

package images;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.Image;

import java.awt.image.BufferedImage;

import javax.swing.ImageIcon;

import ultilityTools.ImageTools;

/\*\*

\* Img is a class that represents an image , which is used to make drawing images a lot easier.

\*

\* @author user

\*/

public class Img {

private Image \_image;

private int x, y, width, height;

public Img() {

\_image = null;

x = y = width = height = 0;

}

/\*\*

\* Constructor

\*

\* @param path

\* - the path string of this Img

\* @param x

\* - the X position of the img

\* @param y

\* - the Y position of the img;

\* @param width

\* - the width of the img;

\* @param height

\* - the height of the img;

\*/

public Img(String path, int x, int y, int width, int height) {

/\*\*

\* gets the image from the resource folder that is located by using the path given

\*/

\_image = new ImageIcon(this.getClass().getClassLoader().getResource(path)).getImage();

setImgCords(x, y);

setImgSize(width, height);

}

/\*\*

\* clone is a function that returns a copy of the current Img without damaging the original

\*/

public Img clone() {

Img clon = new Img("", getX(), getY(), getWidth(), getHeight());

clon.setImage(\_image);

return clon;

}

/\*\*

\* Draws image onto graphics , using Graphics2d

\*

\* @param g

\* -Graphics that are used

\*/

public void drawImg(Graphics g) {

Graphics2D g2d = (Graphics2D) g;

// g2d.setColor(new Color(0, 0, 0, 255));

// g2d.fillRect(x, y, width, height);

g2d.drawImage(\_image, x, y, width, height, null);

}

/\*\*

\* Draws image onto the graphics but in a vertical position

\*

\* @param g

\* - Graphics that are used

\*/

public void drawImgVertically(Graphics g) {

Graphics2D g2d = (Graphics2D) g;

g2d.drawImage(\_image, x + width, y, -width, height, null);

}

/\*\*

\* Draws image onto the graphics but in a horizontal position

\*

\* @param g

\* - Graphics that are used

\*/

public void drawImgHorizontally(Graphics g) {

Graphics2D g2d = (Graphics2D) g;

g2d.drawImage(\_image, x, y + height, width, -height, null);

}

/\*\*

\* Draws image onto the graphics but in an angle rotation clockwise

\*

\* @param g

\* - Graphics that are used

\*/

public void drawImgRotate(Graphics g, double angle) {

setImage(ImageTools.rotate(this, angle));

drawImg(g);

}

/\*\*

\* Draws image onto the graphics but only a cropped part of it

\* @param g - Graphics that are used

\* @param newX - the new X position from the new cropped img

\* @param newY - the new Y position from the new cropped img

\* @param newHeight - the new Height for the new cropped img

\* @param newWidth - the new Width for the new cropped img

\*/

public void drawPartImage(Graphics g, int newX, int newY, int newHeight, int newWidth) {

BufferedImage bimage = ImageTools.crop(this, newX, newY, newHeight, newWidth);

Graphics2D g2d = (Graphics2D) g;

g2d.drawImage(bimage, x + newX, y + newY, newWidth, newHeight, null);

}

/\*\*

\* Draws image onto the graphics but only a cropped part of it and draw it vertically

\* @param g - Graphics that are used

\* @param newX - the new X position from the new cropped img

\* @param newY - the new Y position from the new cropped img

\* @param newHeight - the new Height for the new cropped img

\* @param newWidth - the new Width for the new cropped img

\*/

public void drawPartImageVeritcally(Graphics g, int newX, int newY, int newHeight, int newWidth) {

BufferedImage bimage = ImageTools.crop(this, newX, newY, newHeight, newWidth);

Graphics2D g2d = (Graphics2D) g;

g2d.drawImage(bimage, x + newX + newWidth, y + newY, -newWidth, newHeight, null);

}

/\*\*

\* Sets image cordinates

\*

\* @param x

\* - the X position of the img

\* @param y

\* - the Y position of the img;

\*/

public void setImgCords(int x, int y) {

this.x = x;

this.y = y;

}

/\*\*

\* @param width

\* - the width of the img;

\* @param height

\* - the height of the img;

\*/

public void setImgSize(int width, int height) {

this.width = width;

this.height = height;

}

/\*\*

\* @return the Image of the Img

\*/

public Image getImage() {

return \_image;

}

/\*\*

\* set a new Image of the Img

\* @param image - the new Image

\*/

public void setImage(Image image) {

\_image = image;

}

/\*\*

\* set a new Image of the Img

\* @param path - the path to a picture in the resource folder.

\*/

public void setImage(String path) {

// \_image = new Image

\_image = new ImageIcon(this.getClass().getClassLoader().getResource(path)).getImage();

}

/\*\*

\* @return get X position of the Img

\*/

public int getX() {

return x;

}

/\*\*

\* @return get Y position of the Img

\*/

public int getY() {

return y;

}

/\*\*

\* @return get the Width of the Img

\*/

public int getWidth() {

return width;

}

/\*\*

\* set a new Width for the Img

\* @param width - the new width

\*/

public void setWidth(int width) {

this.width = width;

}

/\*\*

\* @return the height of the Img

\*/

public int getHeight() {

return height;

}

/\*\*

\* set a new Height for the Img

\* @param height - the new height

\*/

public void setHeight(int height) {

this.height = height;

}

}

**mainInitialize.BackgroundImgPanel**

package mainInitialize;

import images.Img;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import javax.imageio.ImageIO;

import javax.swing.BorderFactory;

import javax.swing.BoxLayout;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JLabel;

import javax.swing.JPanel;

import ultilityTools.ImageTools;

import unit.customizeScreen.CustomizeFrame;

import unit.customizeScreen.DisplayScreen;

import unit.player.Player;

import map.MapGFrame;

/\*\*

\* this class represents the panel of mainMenu extends JPanel because it is panel

\*

\* @author Peter

\* @category Singleton - only contains one instance

\*/

public class BackgroundImgPanel extends JPanel {

// The buttons

private JButton \_startBtn;

private JButton \_exitBtn;

private JButton \_settingsBtn;

private JButton \_instructionBtn;

// the images

private Img \_gameLogo;

// gif related

private JLabel \_gifPlayer;

private ImageIcon \_imgIcon;

// The array of available gifs

private String \_gifScenePath[] = { "Menu\\reverseScene1.gif", "Menu\\reverseScene2.gif",

"Menu\\reverseScene3.gif" };

// current gif index

private static int \_currentGifIndex = 0;

// the button panel

private static JPanel \_buttonPanel;

// The array of available buttonIcons

// if the item uses getResource , srcImages is not required at start

private String \_buttonImgPath[] = { "srcImages\\Menu\\startIcon.png", "srcImages\\Menu\\settingsIcon.png",

"srcImages\\Menu\\exitIcon.png", "srcImages\\Menu\\continueIcon.png" ,"srcImages\\Menu\\instructionsIcon.png"};

/\*\* the instance of the game \*/

private static BackgroundImgPanel singleton = new BackgroundImgPanel(MainMenu.getInstance().getGifPlayer());

/\*\*

\* constructor - creates the buttons assigns button listeners and advances the gifPlayer if a

\* new game button is clicked

\*

\* @param main

\* - the frame from which the backgroundImgPanel was called

\* @param gifPlayer

\* - the label in which gifs will be displayed

\*/

private BackgroundImgPanel(JLabel gifPlayer) {

\_gifPlayer = gifPlayer;

new Img("Menu\\menuBackground.png", 0, 0, MainMenu.getInstance().getWidth(),

MainMenu.getInstance().getHeight());

\_gameLogo = new Img("Menu\\gameLogo.png", 0, 0, 264, 100);

\_buttonPanel = new JPanel();

\_startBtn = ImageTools.makeButton(\_buttonImgPath[0]);

\_settingsBtn = ImageTools.makeButton(\_buttonImgPath[1]);

\_instructionBtn = ImageTools.makeButton(\_buttonImgPath[4]);

\_exitBtn = ImageTools.makeButton(\_buttonImgPath[2]);

\_startBtn.addActionListener(new ActionListener() {

/\*\*

\* the startButton action listener. increases the number on the currentGif and changes

\* it initiates the game or shows it if it was hidden sets the frame invisible

\*/

public void actionPerformed(ActionEvent e) {

MainMenu.getInstance().setVisible(false);

++\_currentGifIndex;

\_currentGifIndex %= \_gifScenePath.length;

\_imgIcon = new ImageIcon(

singleton.getClass().getClassLoader().getResource(\_gifScenePath[\_currentGifIndex]));

\_gifPlayer.setIcon(\_imgIcon);

/\*

\* if (GameStats.getInstance().getPlayers().size() == 0) {

\* Main.initPlayers();//initiates positions for players } if

\* (CustomizeFrame.getInstance() != null) { ((Player)

\* GameStats.getPlayers().get(0)).setCustomGraphics(64, 64);//set default custom

\* graphics }

\*/

if (MapGFrame.getInstance() == null) {// if map doesnt exist

GameStats.restartPositions();

MapGFrame.init("Game");

BufferedImage buttonIcon = null;

try {

buttonIcon = ImageIO.read(new File(\_buttonImgPath[3]));

\_startBtn.setIcon(new ImageIcon(buttonIcon));

} catch (IOException e1) {

}

} else {// if map exists

MapGFrame.getInstance().setVisible(true);

}

GameStats.startGame();

}

});

\_instructionBtn.addActionListener(new ActionListener(){

@Override

public void actionPerformed(ActionEvent e) {

\_buttonPanel.setVisible(false);

add(InstructionPanel.init());

InstructionPanel.getInstance().setVisible(true);

}

});

\_settingsBtn.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

MainMenu.getInstance().setVisible(false);

if (CustomizeFrame.getInstance() == null)

CustomizeFrame.init();

else {

CustomizeFrame.getInstance().setVisible(true);

}

DisplayScreen.copyToDisplay();

}

});

\_exitBtn.addActionListener(new ActionListener() {

/\*\* the exitButton action listener. exits the system \*/

public void actionPerformed(ActionEvent e) {

System.exit(0);

}

});

\_buttonPanel.setLayout(new BoxLayout(\_buttonPanel, BoxLayout.Y\_AXIS));

BufferedImage img = ImageTools.toBufferedImage(\_gameLogo);

\_buttonPanel.add(new JLabel(new ImageIcon(img)));

\_buttonPanel.setBackground(Color.black);

\_buttonPanel.add(\_startBtn);

\_buttonPanel.add(\_instructionBtn);

\_buttonPanel.add(\_settingsBtn);

\_buttonPanel.add(\_exitBtn);

\_buttonPanel.setOpaque(true);

setLayout(new BorderLayout());

add(\_buttonPanel, BorderLayout.EAST);

}

/\*\*

\* get the Singleton instance

\*

\* @return BackgroundImgPanel only instance

\*/

public static BackgroundImgPanel getInstance() {

return singleton;

}

public static JPanel getButtonPanel(){

return \_buttonPanel;

}

}

**mainInitialize.GameStats**

package mainInitialize;

import java.awt.Point;

import java.util.LinkedList;

import engine.Block;

import engine.dijekstra.Dijekstra;

import engine.input.KeyboardInput;

import map.MapGFrame;

import map.MapGPanel;

import unit.MovingUnit;

import unit.WinningInterface;

import unit.customizeScreen.CustomizeFrame;

import unit.enemy.Enemy;

import unit.player.Player;

/\*\*

\* Gamestats is a class that represents a global overview on the game's statistics as well

\* containing information about every object on the map.

\*

\* @author Peter

\* @category Singleton - only contains one instance

\*/

public class GameStats {

private static LinkedList<MovingUnit> \_players;

private static LinkedList<Block> \_fullWalls;

private static LinkedList<Block> \_halfWalls;

private static LinkedList<Point> \_playerOccupied;

// private static GameStats singleton = new GameStats();

public final static int \_height = 15;

public final static int \_width = 15;

private static int \_playerWon=0;

private static int \_enemyWon=0;

/\*\*

\* Constructor is private because of singleton

\*/

private GameStats() {

\_players = new LinkedList<MovingUnit>();

\_fullWalls = new LinkedList<Block>();

\_halfWalls = new LinkedList<Block>();

\_playerOccupied = new LinkedList<Point>();

}

/\*\*

\* adds a unit to the MovingUnit list

\*

\* @param player

\* - the unit we want to add

\*/

public static void addPlayer(MovingUnit player) {

\_players.add(player);

}

/\*\*

\* @return the linkedlist with all MovingUnit

\*/

public static LinkedList<MovingUnit> getPlayers() {

return \_players;

}

/\*\*

\* add fullWall block to the fullwall list

\*

\* @param fullWall

\* - the Block we want to add

\*/

public static void addFullWall(Block fullWall) {

\_fullWalls.add(fullWall);

}

/\*\*

\* @return the linkedlist with all FullWall blocks

\*/

public static LinkedList<Block> getFullWalls() {

return \_fullWalls;

}

/\*\*

\* removes fullwall block from the list (uses comparison based on addresses)

\*

\* @param fullWall

\* - the fullWall block we want to remove from the list

\* @return true if successfuly removed the block, else false.

\*/

public static boolean removeFullWall(Block fullWall) {

if (\_fullWalls.contains(fullWall)) {

return \_fullWalls.remove(fullWall);

}

return false;

}

/\*\*

\* add halfWall block to the halfwall list

\*

\* @param halfWall

\* - the Block we want to add

\*/

public static void addHalfWall(Block halfWall) {

\_halfWalls.add(halfWall);

}

/\*\*

\* @return the linkedlist with all HalfWall blocks

\*/

public static LinkedList<Block> getHalfWalls() {

return \_halfWalls;

}

/\*\*

\* removes halfwall block from the list (uses comparison based on addresses)

\*

\* @param halfWall

\* - the halfWall block we want to remove from the list

\* @return true if successfuly removed the block, else false.

\*/

public static boolean removeHalfWall(Block halfWall) {

if (\_halfWalls.contains(halfWall)) {

return \_halfWalls.remove(halfWall);

}

return false;

}

/\*\*

\* add occupiedPoint block to the playerOccupied list

\*

\* @param p

\* - the point we want to add

\*/

public static void addOccupied(Point p) {

if (!\_playerOccupied.contains(p))

\_playerOccupied.add(p);

}

/\*\*

\* add occupiedPoint block to the playerOccupied list

\*

\* @param i

\* - the X coordinate of the point

\* @param j

\* - the Y coordinate of the point

\*/

public static void addOccupied(int i, int j) {

addOccupied(new Point(i, j));

}

/\*\*

\* remove occupied Point from the playerOccupied list

\*

\* @param i

\* - the X coordinate of the point we want to remove

\* @param j

\* - the Y coordinate of the point we want to remove

\* @return true if removed successfuly else false.

\*/

public static boolean removeOccupied(int i, int j) {

if (occupiedExists(i, j))

return \_playerOccupied.remove(new Point(i, j));

return false;

}

/\*\*

\* check if point with similar coordinates exists in playerOccupied List

\*

\* @param i

\* - the X coordinate of the point

\* @param j

\* - the Y coordinate of the point

\* @return

\*/

public static boolean occupiedExists(int i, int j) {

return (\_playerOccupied.contains(new Point(i, j)));

}

/\*\*

\* calls pauseUnit for all moving units that are registered in the Global MovingUnit list.

\*/

public static void pauseGame() {

for (MovingUnit mu : \_players) {

((WinningInterface) mu).pauseUnit();

}

}

/\*\*

\* calls startUnit for all moving units that are registered in the Global MovingUnit list.

\*/

public static void startGame() {

for (MovingUnit mu : \_players) {

((WinningInterface) mu).startUnit();

}

}

/\*\*

\* Creates a new MapGPanel which generates a completly new map

\* @param newMap - if we want to create a new map or just restart the previous one

\*/

public static void generateNewPanel(boolean newMap) {

if (MapGPanel.getInstance() != null) {

MapGFrame.getInstance().remove(MapGPanel.getInstance());

}

MapGFrame.getInstance().add(MapGPanel.init(15, 15, 64, newMap));

}

/\*\*

\* empties all of the global lists and initiates all positions of the Units

\* if there is no custom graphics its sets a default

\*/

public static void restartPositions() {

new GameStats();

Main.initPlayers();

if (CustomizeFrame.getInstance() != null) {

((Player) GameStats.getPlayers().get(0)).setCustomGraphics(64, 64);// set default custom

// graphics

}

}

/\*\*

\* adds the players' panel into the panel of the map

\*/

public static void addUnitPanels() {

for (MovingUnit mu : GameStats.getPlayers()) {

MapGPanel.getInstance().add(mu);

}

}

/\*\*

\* Restarts the entire game with a choise of creating a new map or keeping the last one.

\* @param newMap-if true , it creates a new map else it keeps the last one

\*/

public static void restartGame(boolean newMap) {

// KeyboardInput.getInstance().cleanListenerList();

KeyboardInput.getInstance().resetAllKeys();

restartPositions();

generateNewPanel(newMap);

addUnitPanels();

pauseGame();

MapGFrame.getInstance().setVisible(false);

MapGFrame.getInstance().setVisible(true);

MapGFrame.getInstance().setFocusable(true);

}

/\*\*

\* speeding up the enemy's timer and slowing down the player's timer

\*/

public static void increaseDifficaulty() {

if (Enemy.\_timerTicker > 5)

Enemy.\_timerTicker -= 2;

if (Player.\_timerTicker < 100)

Player.\_timerTicker += 2;

}

/\*\*

\* slowing down the enemy's timer and speeding up the player's timer

\*/

public static void decreaseDifficaulty() {

if (Enemy.\_timerTicker < 70)

Enemy.\_timerTicker += 3;

if (Player.\_timerTicker > 4)

Player.\_timerTicker -= 2;

}

/\*\*

\* increase by 1 the matches won by the player

\*/

public static void playerWonInc(){

\_playerWon++;

}

/\*\*

\* increase by 1 the matches won by the enemy

\*/

public static void enemyWonInc(){

\_enemyWon++;

}

/\*\*

\* @return the matches won by player

\*/

public static int getPlayerWon() {

return \_playerWon;

}

/\*\*

\* @return the matches won by enemy

\*/

public static int getEnemyWon() {

return \_enemyWon;

}

}

**mainInitialize.Main**

package mainInitialize;

import unit.enemy.Enemy;

import unit.player.Player;

/\*\*

\* The main class represents the starter of the game

\* @author Peter

\*

\*/

public class Main {

/\*\*

\* the main function , runs first initiates the game

\*/

public static void main(String[] args) {

MainMenu.init();

}

/\*

\* List of singletons

\* MainMenu

\* BackgroundImgPanel

\* MapGFrame

\* MapGPanel

\* MapG

\* CustomizeFrame

\* CustomizeScreen

\* DisplayScreen

\*/

/\*

\* List of objectives:

\* Diffrent Floor\ Wall layout 1 V

\* Power ups

\* Health points 3 [ finished game]

\* Win\Lose 2

\* New Projectiles

\* Diffrent Enemies

\* Diffrent Map Layouts

\* Classes

\*/

/\*\*

\* adds the players to the global player list and sets their places as occupied

\*/

public static void initPlayers(){

GameStats.addPlayer(new Player(64\*1, 64\*1, 64, 64));

GameStats.addOccupied(1, 1);

GameStats.addPlayer(new Enemy(64\*13, 64\*13, 64, 64));

GameStats.addOccupied(13, 13);

}

}

**mainInitialize.MainMenu**

package mainInitialize;

import java.awt.BorderLayout;

import javax.swing.ImageIcon;

import javax.swing.JFrame;

import javax.swing.JLabel;

import engine.Unit;

/\*\*

\* MainMenu is a class that represents a main menu screen ,extends JFrame because it is frame

\*

\* @author Peter

\* @category Singleton - only contains one instance

\*/

public class MainMenu extends JFrame {

private static MainMenu singleton = null;

private Unit Units[];

private BackgroundImgPanel \_backGroundPanel;

private JLabel \_gifPlayer;

private ImageIcon \_imgIcon;

/\*\*

\* constructor is private because of singleton, puts the first gif and starts backgroundImgPanel

\*/

private MainMenu() {

if (singleton == null) {

singleton = this;

setSize(605, 338);

\_imgIcon = new ImageIcon(this.getClass().getClassLoader().getResource("Menu\\reverseScene1.gif"));

\_gifPlayer = new JLabel(\_imgIcon);

\_gifPlayer.setBounds(0, 0, 600, 338);

\_imgIcon.setImageObserver(\_gifPlayer);

add(\_gifPlayer, BorderLayout.WEST);

add(BackgroundImgPanel.getInstance());

setResizable(false);

setVisible(true);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

}

}

/\*\*

\* @return the JLabel that plays the gifs

\*/

public JLabel getGifPlayer() {

return \_gifPlayer;

}

/\*\*

\* initialize the singleton instance

\*

\* @return MainMenu only instance

\*/

public static MainMenu init() {

if (singleton == null)

singleton = new MainMenu();

return singleton;

}

/\*\*

\* get the Singleton instance

\*

\* @return MainMenu only instance

\*/

public static MainMenu getInstance() {

return singleton;

}

}

**map.EscapeButtonInterface**

package map;

/\*\*

\* Interface that should be used when 'Escape' button is pressed

\* @author Peter

\*

\*/

public interface EscapeButtonInterface {

void EscapeButtonClicked();

}

**map.MapG**

package map;

import java.io.File;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.Document;

import org.w3c.dom.NamedNodeMap;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import engine.Block;

/\*\*

\* MapG is a class that represents the graphical structure of the map

\*

\* @author Peter

\* @category Singleton - only contains one instance

\*/

public class MapG {

private int \_size;

private int \_counter = 0;

// private int[][] \_map;

private static Block[][] \_map;

private static MapG singleton = null;

// size - Height

// sizeW - Width

/\*\*

\* Constructor reads if the xml path given has childnodes and fills map with the blocks

\*

\* @param size

\* - the row amount

\* @param sizeW

\* - the col amount

\* @param fileName

\* - the file path

\*/

private MapG(int size, int sizeW, String wallName, String floorName) {

// Map is a mat sizeW\*size

\_map = new Block[size][sizeW];

\_size = sizeW;

try {

File wallFile = new File(wallName);

File floorFile = new File(floorName);

DocumentBuilder docBuilder = DocumentBuilderFactory.newInstance().newDocumentBuilder();

Document wallDoc = docBuilder.parse(wallFile);

Document floorDoc = docBuilder.parse(floorFile);

if (wallDoc.hasChildNodes() && floorDoc.hasChildNodes()) {

readNode(wallDoc.getChildNodes(), floorDoc.getChildNodes());

}

} catch (Exception e) {

System.out.println(e.getMessage());

}

}

/\*\*

\* initialize the singleton instance

\*

\* @return MapG only instance

\*/

public static MapG init(int size, int sizeW, String wallName, String floorName) {

if (singleton == null)

singleton = new MapG(size, sizeW, wallName, floorName);

return singleton;

}

/\*\*

\* get the Singleton instance

\*

\* @return MapG only instance

\*/

public static MapG getInstance() {

return singleton;

}

/\*\*

\* @return Block Matrix that is read from the XML

\*/

public static Block[][] get\_map() {

return \_map;

}

/\*\*

\* the function reads from the nodeList (xml) the values of the attributes and updates \_map

\* based on it from both floorList and wallList

\*

\* @param wallList - the nodes that contain walls

\* @param floorList - the nodes that contain floors

\*/

private void readNode(NodeList wallList, NodeList floorList) {

for (int count = 0; count < wallList.getLength(); count++) {

Node tempWallNode = wallList.item(count);

Node tempFloorNode = floorList.item(count);

if (tempWallNode.getNodeType() == Node.ELEMENT\_NODE) {

if (tempWallNode.hasAttributes()) {

NamedNodeMap nodeWallMap = tempWallNode.getAttributes();

NamedNodeMap nodeFloorMap = tempFloorNode.getAttributes();

for (int i = 0; i < nodeWallMap.getLength(); i++) {

Node wallNode = nodeWallMap.item(i);

Node floorNode = nodeFloorMap.item(i);

\_map[\_counter % \_size][\_counter / \_size] = new Block(Integer.parseInt(wallNode.getNodeValue()),

\_counter % \_size, \_counter / \_size, 64, 64);

\_map[\_counter % \_size][\_counter / \_size]

.setFloorIndex(((Integer.parseInt(floorNode.getNodeValue())) >> 4) & 0b1111);

// System.out.println(" I:" + \_counter / \_size + " J:" + \_counter % \_size +

// " "

// + \_map[\_counter / \_size][\_counter % \_size]);

\_counter++;

}

}

if (tempWallNode.hasChildNodes()) {

readNode(tempWallNode.getChildNodes(), tempFloorNode.getChildNodes());

}

}

}

}

}

**map.MapGFrame**

package map;

import mainInitialize.GameStats;

import mainInitialize.MainMenu;

import unit.player.Player;

import unit.player.PlayerG;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.GridBagLayout;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JTextField;

import engine.input.KeyboardInput;

/\*\*

\* MapGFrame is a class that represents the frame of the map extends JFrame because it is frame ,

\* implements EscapeButtonInterface because escape calls main menu

\*

\* @author Peter

\* @category Singleton - only contains one instance

\*/

public class MapGFrame extends JFrame implements EscapeButtonInterface {

private static MapGFrame singleton = null;

/\*\*

\* Constructor

\*

\* @param frameTitle-

\* the title of the frame

\*/

private MapGFrame(String frameTitle) {

super(frameTitle);

/\*\* adds to the keyboardInput it self as listener \*/

KeyboardInput.getInstance().addListener(this);

/\*\* creates graphical map \*/

// setLayout(new GridBagLayout());

add(MapGPanel.init(15, 15, 64));

/\*\* creates player \*/

for (int i = 0; i < GameStats.getPlayers().size(); i++) {

MapGPanel.getInstance().add(GameStats.getPlayers().get(i));

}

// MapGPanel.getInstance().add(new Player(128, 64\*3, 64, 64));

// setExtendedState(JFrame.MAXIMIZED\_BOTH);

// setUndecorated(true);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(15 \* (64 + 1), 15 \* (64 + 2) + 8);

setVisible(true);

}

/\*\*

\* initialize the singleton instance

\*

\* @return MapGFrame only instance

\*/

public static MapGFrame init(String frameTitle) {

if (singleton == null)

singleton = new MapGFrame(frameTitle);

return singleton;

}

/\*\*

\* get the Singleton instance

\*

\* @return MapGFrame only instance

\*/

public static MapGFrame getInstance() {

return singleton;

}

/\*\*

\* the implemented method is called when 'escape' is clicked

\*/

@Override

public void EscapeButtonClicked() {

MainMenu.getInstance().setVisible(true);

GameStats.pauseGame();

setVisible(false);

// KeyboardInput.getInstance().setValue(\_ind, value);

}

}

**map.MapGPanel**

package map;

import java.awt.Color;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.GridBagLayout;

import java.awt.Point;

import java.awt.image.BufferedImage;

import java.util.LinkedList;

import javax.swing.JButton;

import javax.swing.JPanel;

import javax.swing.JTextField;

import engine.Block;

import engine.RandomMapCreator;

import engine.dijekstra.Dijekstra;

import engine.dijekstra.EdgeInt;

import engine.dijekstra.Graph;

import images.Img;

import mainInitialize.GameStats;

import ultilityTools.ImageTools;

import unit.MovingUnit;

import unit.ScorePanel;

import unit.enemy.Enemy;

import unit.player.Player;

/\*\*

\* MapPanel is a class that represents a map's panel

\*

\* @author Peter

\* @category Singleton - only contains one instance

\*/

public class MapGPanel extends JPanel {

private int \_mapHeight;

private int \_mapWidth;

private int \_blockSize;

private boolean first = false;

private Graph graph;

public LinkedList<Point> list = new LinkedList<Point>();

/\*\* All Available walls \*/

private String \_wallPaths[] = { "Walls\\0001Wall\_Atlas.png", "Walls\\0010WallLog\_Atlas.png",

"Walls\\0011Palisade\_Atlas.png", "Walls\\0100Stone\_Atlas.png", "Walls\\0101Fence\_Atlas.png",

"Walls\\0110Parapet\_Atlas.png", "Walls\\0111SilverWall\_Atlas.png", "Walls\\1000GlassWall\_Atlas.png",

"Walls\\1001UWall\_Atlas.png", "Walls\\1010UWallA\_Atlas.png", "Walls\\1011UWallB\_Atlas.png",

"Walls\\1100UWallYellow\_Atlas.png", "Walls\\1101UWallYellowOff\_Atlas.png",

"Walls\\1110BrickWall\_Atlas.jpg" };

/\*\* All wall destructions \*/

private String \_destructionPaths[] = { "Walls\\Animations\\wallDestruction\_6.png",

"Walls\\Animations\\wallDestruction\_5.png", "Walls\\Animations\\wallDestruction\_4.png",

"Walls\\Animations\\wallDestruction\_3.png", "Walls\\Animations\\wallDestruction\_2.png",

"Walls\\Animations\\wallDestruction\_1.png" };

/\*\* All Available floors \*/

private String \_floorPaths[] = { "Floors\\0001AluminiumTile.png", "Floors\\0010CarpetBeige.png",

"Floors\\0011CarpetBlack.png", "Floors\\0100CarpetBlue.png", "Floors\\0101CarpetGreen.png",

"Floors\\0110CarpetRed.png", "Floors\\0111CarpetWhite.png", "Floors\\1000CopperTile.png",

"Floors\\1001FloorFI.png", "Floors\\1010WoodFloor1.png", "Floors\\1011WoodFloor2.png",

"Floors\\1100WoodFloor3.png", "Floors\\1101WoodFloor4.png", "Floors\\1110WoodFloor5.png",

"Floors\\1111WoodFloor6.png" };

private LinkedList<Img> \_wall;

private LinkedList<Img> \_floor;

private LinkedList<Img> \_destruction;

private static MapGPanel singleton = null;

/\*\*

\* constructor prepares the \_wall and \_floor and \_destruction array , also creates a new map

\* from the two XML files

\*

\* @param mapHeight

\* - the height of the map

\* @param mapWidth

\* - the width of the map

\* @param blockSize

\* - the size of a block

\*/

private MapGPanel(int mapHeight, int mapWidth, int blockSize,boolean createNewMap) {

//setBackground(Color.BLACK);

//setOpaque(true);

\_mapHeight = mapHeight;

\_mapWidth = mapWidth;

\_blockSize = blockSize;

if(createNewMap)

new RandomMapCreator(mapHeight, mapWidth);

MapG.init(\_mapHeight, \_mapWidth, "mapTemplates\\randomMapWall.xml", "mapTemplates\\randomMapFloor.xml");

\_wall = new LinkedList<Img>();

\_floor = new LinkedList<Img>();

\_destruction = new LinkedList<Img>();

for (int i = 0; i < \_wallPaths.length; i++) {

addSplitAtlasAddToList(\_wallPaths[i], i);

}

for (int i = 0; i < \_floorPaths.length; i++) {

\_floor.add(new Img(\_floorPaths[i], 0, 0, \_blockSize, \_blockSize));

}

for (int i = 0; i < \_destructionPaths.length; i++) {

\_destruction.add(new Img(\_destructionPaths[i], 0, 0, \_blockSize, \_blockSize));

}

setLayout(null);

}

/\*\*

\* initialize the singleton instance

\*

\* @return MapGPanel only instance

\*/

public static MapGPanel init(int mapHeight, int mapWidth, int blockSize,boolean createNewMap) {

//if (singleton == null)

singleton = new MapGPanel(mapHeight, mapWidth, blockSize,createNewMap);

return singleton;

}

/\*\*

\* get the Singleton instance

\*

\* @return MapGPanel only instance

\*/

public static MapGPanel getInstance() {

return singleton;

}

/\*\*

\* the function receives an atlas of walls(sprites) and creates separate images and puts them

\* into \_walls

\*

\* @param imagePath

\* - the path to the atlas

\* @param wallsInd

\* - the index of the current atlas

\*/

public void addSplitAtlasAddToList(String imagePath, int wallsInd) {

Img atlas = new Img(imagePath, 0, 0, 256, 256);

Img small = new Img("", 0, 0, 64, 64);

BufferedImage bi = ImageTools.toBufferedImage(atlas);

for (int index = 0; index < 16; index++) {

small.setImage((bi.getSubimage(index % 4 \* \_blockSize, index / 4 \* \_blockSize, small.getHeight(),

small.getWidth())));

// \_wall.addLast(small.clone());

\_wall.add(wallsInd \* 16 + index, small.clone());

}

}

/\*\*

\* the paintComponent of the map

\*/

public void paintComponent(Graphics g) {

super.paintComponent(g);

// System.out.println("Paint Comp");

int modelIndex, bonusModel, damageIndex;

Block temp;

for (int i = 0; i < \_mapHeight; i++) {

for (int j = 0; j < \_mapWidth; j++) {

temp = MapG.getInstance().get\_map()[j][i];

/\*\* compare the wall level , if 0 then floor \*/

if (temp.getWallLevel() == 0) {// floors

/\*\* get the modelIndex of the block \*/

modelIndex = temp.getModelIndex();

/\*\* set coordinates \*/

\_floor.get(modelIndex).setImgCords((j \* \_blockSize), (i \* \_blockSize));

\_floor.get(modelIndex).drawImg(g);

}

/\*\* compare the wall level , if 1 or above then wall \*/

if (temp.getWallLevel() >= 1) {// walls

/\*\* get the modelIndex of the block \*/

modelIndex = temp.getModelIndex();

/\*\* check if bonus models are enabled \*/

if ((temp.getImgID() & 0b0100) == 0b0100) {// desigenated wall

/\*\* get the bonusModel of the block \*/

bonusModel = temp.getBonusModel();

/\*\* default floor \*/

\_floor.get(temp.getFloorModelIndex()).setImgCords((j \* \_blockSize), (i \* \_blockSize));

\_floor.get(temp.getFloorModelIndex()).drawImg(g);

/\*\* set coordinates \*/

\_wall.get(modelIndex \* 16 + bonusModel).setImgCords((j \* \_blockSize), (i \* \_blockSize));

\_wall.get(modelIndex \* 16 + bonusModel).drawImg(g);

} else {// default

/\*\* default bonus model \*/

bonusModel = 12;

\_floor.get(0).setImgCords((j \* \_blockSize), (i \* \_blockSize));

\_floor.get(0).drawImg(g);

/\*\* set coordinates \*/

\_wall.get(modelIndex \* 16 + bonusModel).setImgCords((j \* \_blockSize), (i \* \_blockSize));

\_wall.get(modelIndex \* 16 + bonusModel).drawImg(g);

}

if (!temp.isFull()) {// half wall

// draw damaged model

damageIndex = getDamageIndex(temp);

if (damageIndex < \_destructionPaths.length) {

\_destruction.get(damageIndex).setImgCords((j \* \_blockSize), (i \* \_blockSize));

\_destruction.get(damageIndex).drawImg(g);

}

}

}

}

}

drawHUD(g);

/\*

\* Links if(!first){ graph = new Graph(\_mapHeight, \_mapWidth);//((Enemy)

\* GameStats.getPlayers().get(1)).getMover().getGraph(); graph.buildGraph(); first=true; }

\* int center = 64 / 2; int colorR = 63, colorL = 190, colorU = 63, colorD = 190;

\* //g.setColor(Color.BLACK); //for (int i = 0; i < \_mapHeight; i++) //for (int j = 0; j <

\* \_mapWidth; j++) //g.fillOval((i + 1) \* 64 - 48, (j + 1) \* 64 - 48, 32, 32); for (int i =

\* 0; i < \_mapHeight-1; i++) { for (int j = 0; j < \_mapWidth; j++) { for (EdgeInt e :

\* graph.getGraph()[i][j].getAdjacencies()) { //System.out.println("-["+i+", "+j+"] -> [" +

\* e.getTarget().getI() + ", " + e.getTarget().getJ() + "]"); if (e.getTarget().getI() == i)

\* { if (e.getTarget().getJ() == j + 1) {// right // g.setColor(new Color(colorR, 127, 127,

\* 200)); g.setColor(Color.green); g.fillRect((j + 1) \* 64 - center, (i + 1) \* 64 - center -

\* 3, 60, 6); } else {// left // g.setColor(new Color(colorL, 127, 127, 200));

\* g.setColor(Color.yellow); g.fillRect((j + 1) \* 64 - center, (i + 1) \* 64 - center + 3,

\* -60, 6); } } else if (e.getTarget().getI() == i + 1) {// up g.setColor(Color.red); //

\* g.setColor(new Color(127, 127, colorD, 200)); g.fillRect((j + 1) \* 64 - center - 3, (i +

\* 1) \* 64 - center, 6, 60); } else {// down // g.setColor(new Color(127, 127, colorU,

\* 200)); g.setColor(Color.blue); g.fillRect((j + 1) \* 64 - center + 3, (i + 1) \* 64 -

\* center, 6, 60); } } } }

\*/

/\*

\* g.setColor(new Color(55, 55, 55, 120)); for (Point p : list) { g.fillRect((int) p.getX(),

\* (int) p.getY(), 64, 64); }

\*/

}

/\*\*

\* draws the health and scores of players and enemies at the top of the map

\* @param g

\*/

private void drawHUD(Graphics g){

g.setFont(new Font("My Font", Font.BOLD, 16));

for (MovingUnit mu : GameStats.getPlayers()) {

if (mu instanceof Player) {

g.setColor(new Color(255,50,50,200));

g.drawString("HEALTH:" + ((int)(mu.getHealthPoints()\*100)) + " %", 125, 40);

} else {

g.setColor(new Color(50,50,255,200));

g.drawString("HEALTH:" + ((int)(mu.getHealthPoints()\*100)) + " %", 725, 40);

}

}

g.setColor(new Color(255,50,50,200));

g.drawString("SCORE:" + Integer.toString(GameStats.getPlayerWon()), 45, 40);

g.setColor(new Color(50,50,255,200));

g.drawString("SCORE:" + Integer.toString(GameStats.getEnemyWon()),845 , 40);

}

/\*\*

\* get the index for Destruction array based on the current health state of the block

\*

\* @param block

\* - the Block we want to get index for

\* @return number below \_destructionPaths.length if there is a proper index , else return the

\* \_destructionPaths.length

\*/

private int getDamageIndex(Block block) {

int result = (int) (block.getHpDivision() \* \_destructionPaths.length);

return (result < 0) ? \_destructionPaths.length : result;

}

}

**ultilityTools.FileChooser**

package ultilityTools;

import java.io.File;

import javax.swing.JFileChooser;

import javax.swing.filechooser.FileFilter;

/\*\*

\* FileChooser is a class that represents a file browser extends JFileChooser

\* because it is choosing

\*

\* @author Peter

\*

\*/

public class FileChooser extends JFileChooser {

private JFileChooser \_jfc;

private String[] \_filter = { ".png" };

/\*\*

\* Constructor initiates the JFileChooser

\*/

public FileChooser() {

\_jfc = new JFileChooser();

}

/\*\*

\* choose png file from the given startingFolderPath

\*

\* @param startingFolderPath

\* - the starting browsing point

\* @return the selected file

\*/

public File choosePngFile(String startingFolderPath) {

\_jfc.setFileSelectionMode(JFileChooser.FILES\_ONLY);

\_jfc.setAcceptAllFileFilterUsed(false);

/\*\* set a filter \*/

\_jfc.setFileFilter(new FileFilter() {

@Override

public String getDescription() {

return "Png files from clothing category";

}

@Override

public boolean accept(File f) {

for (int i = 0; i < \_filter.length; i++)

if (f.getName().endsWith(\_filter[i]))

return true;

return f.isDirectory();

}

});

File f = new File(startingFolderPath);

\_jfc.setCurrentDirectory(f);

\_jfc.showOpenDialog(null);

return \_jfc.getSelectedFile();

}

}

**ultilityTools.FileFinder**

package ultilityTools;

import java.io.File;

import java.util.LinkedList;

/\*\*

\* FileFinder is a class that searches files in a certain folder

\* @author Peter

\*

\*/

public class FileFinder {

/\*\*

\* picks all files in the given folder where the name ends with a specific part

\* @param folderPath - the folder path of the wanted folder

\* @param \_endingPart - the specific part to sort by

\* @return

\*/

public static LinkedList<String> getAllFiles(String folderPath,String \_endingPart){

File f = new File(folderPath);

LinkedList<String> finalFiles = new LinkedList<String>();

File listOfFiles[] = f.listFiles();

for(int i=0;i<listOfFiles.length;i++){

if(listOfFiles[i].isFile() && listOfFiles[i].getName().endsWith(\_endingPart))

finalFiles.add(listOfFiles[i].getName());

}

return finalFiles;

}

}

**ultilityTools.ImageTools**

package ultilityTools;

import images.Img;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.Image;

import java.awt.image.BufferedImage;

import javax.swing.JPanel;

/\*\*

\* Image tools is a class that represents usefull function that are going to be used throughout the

\* project

\*

\* @author Peter

\*/

public class ImageTools {

/\*\*

\* The function converts img to buffered image since you cant always convert image to

\* BufferedImage you will have to draw a copy of it on the new buffered Image.

\*

\* @param img

\* - the img we want to convert to buffered Image

\* @return - an instance of BufferedImage

\*/

public static BufferedImage toBufferedImage(Img img) {

if (img.getImage() instanceof BufferedImage) {

return (BufferedImage) img.getImage();

}

// Create a buffered image with transparency

BufferedImage bimage = new BufferedImage(img.getWidth(), img.getHeight(), BufferedImage.TYPE\_INT\_ARGB);

// Draw the image on to the buffered image

Graphics2D bGr = bimage.createGraphics();

bGr.drawImage(img.getImage(), 0, 0, null);

bGr.dispose();

// Return the buffered image

return bimage;

}

/\*\*

\* the function draws a panel into Buffered Image which can be used again instead of drawing the

\* components of panel again

\*

\* @param panel

\* @return the new bufferedImage which is the panel

\*/

public static BufferedImage createImageFromPanel(JPanel panel) {

int w = panel.getWidth();

int h = panel.getHeight();

// Create a buffered image with transparency

BufferedImage bi = new BufferedImage(w, h, BufferedImage.TYPE\_INT\_ARGB);

// Draw the panel on to the buffered image

Graphics g = bi.createGraphics();

panel.paint(g);

// Return the buffered image

return bi;

}

/\*\*

\* Rotates an image. Actually rotates a new copy of the image.

\*

\* @param img

\* The image to be rotated

\* @param angle

\* The angle in degrees

\* @return The rotated image

\*/

public static Image rotate(Img img, double angle) {

double sin = Math.abs(Math.sin(Math.toRadians(angle))), cos = Math.abs(Math.cos(Math.toRadians(angle)));

int w = img.getWidth(), h = img.getHeight();

int neww = (int) Math.floor(w \* cos + h \* sin), newh = (int) Math.floor(h \* cos + w \* sin);

BufferedImage bimg = new BufferedImage(w, h, BufferedImage.TYPE\_INT\_ARGB);

Graphics2D g = bimg.createGraphics();

g.translate((neww - w) / 2, (newh - h) / 2);

g.rotate(Math.toRadians(angle), w / 2, h / 2);

g.drawRenderedImage(toBufferedImage(img), null);

g.dispose();

return bimg;

}

/\*\*

\* Crops an Img by the new given coordinates

\* @param imgToCrop - the Img to crop

\* @param cropX - the X position to start the crop from

\* @param cropY - the Y position to start the crop from

\* @param cropHeight - the new Height to start the crop from

\* @param cropWidth - the new Width to start the crop from

\* @return

\*/

public static BufferedImage crop(Img imgToCrop, int cropX, int cropY, int cropHeight, int cropWidth) {

BufferedImage bimage = new BufferedImage(imgToCrop.getWidth(), imgToCrop.getHeight(),

BufferedImage.TYPE\_INT\_ARGB);

Graphics2D bGr = bimage.createGraphics();

bGr.drawImage(imgToCrop.getImage(), 0, 0, imgToCrop.getWidth(), imgToCrop.getHeight(), null);

bGr.dispose();

return bimage.getSubimage(cropX, cropY, cropWidth, cropHeight);

}

}

**ultilityTools.PaintingInterface**

package ultilityTools;

import java.awt.Graphics;

/\*\*

\* every object that wants to be displayed on map must implement this interface

\* @author Peter

\*

\*/

public interface PaintingInterface {

void myPaintComponent(Graphics g);

}

**unit.FireCooldown**

package unit;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

/\*\*

\* Firecooldown is a class that is meant to be the function of a timer which would restrict the unit

\* to use weapons

\*

\* @author Peter

\*/

public class FireCooldown implements ActionListener {

private int \_fireCooldown = 0;

/\*\*

\* the action that is done whenever a timer ticks

\*/

@Override

public void actionPerformed(ActionEvent e) {

if (\_fireCooldown > 0)

\_fireCooldown--;

}

/\*\*

\* @return the fireCooldown countdown

\*/

public int getFireCooldown() {

return \_fireCooldown;

}

/\*\*

\* set the fire cooldown for the countdown

\* @param fireCooldown - the new fire cooldown

\*/

public void setFireCooldown(int fireCooldown) {

\_fireCooldown = fireCooldown;

};

}

**unit.MovingUnit**

package unit;

import javax.swing.Timer;

import mainInitialize.GameStats;

import map.MapG;

import map.MapGPanel;

import unit.enemy.Enemy;

import unit.player.Player;

import unit.player.PlayerG;

/\*\*

\* MovingUnit is a class that represents a moving panel

\*

\* @author Peter

\*/

public abstract class MovingUnit extends ObjectOnMap {

/\*\* the index of the current action ,16 is end \*/

protected int \_i = 0;

protected Timer \_fireTimer;

protected FireCooldown \_fireListener;

/\*\* the timer that does action \*/

protected Timer \_timer;

public MovingUnit(int x, int y, int width, int height) {

super(x, y, width, height);

}

/\*\*

\* moveLeft animation uses \_i as progress does diffrent animations based on isReturning

\*

\* @param isReturning

\* - if true the animation will be a bump else the animation would be a block

\* movement

\*/

public void moveLeft(boolean isReturning) {

if (!isReturning) {

if (\_i < 16) {

setObjectX(getObjectX() - 4);

}

\_i++;

} else {

if (\_i < 2)

setObjectX(getObjectX() - 4);

else if (\_i < 4)

setObjectX(getObjectX() + 4);

if (++\_i == 4) {

\_i = 16;

}

}

}

/\*\*

\* moveRight animation uses \_i as progress does diffrent animations based on isReturning

\*

\* @param isReturning

\* - if true the animation will be a bump else the animation would be a block

\* movement

\*/

public void moveRight(boolean isReturning) {

if (!isReturning) {

if (\_i < 16) {

setObjectX(getObjectX() + 4);

}

\_i++;

} else {

if (\_i < 2)

setObjectX(getObjectX() + 4);

else if (\_i < 4)

setObjectX(getObjectX() - 4);

if (++\_i == 4) {

\_i = 16;

}

}

}

/\*\*

\* moveUp animation uses \_i as progress does diffrent animations based on isReturning

\*

\* @param isReturning

\* - if true the animation will be a bump else the animation would be a block

\* movement

\*/

public void moveUp(boolean isReturning) {

if (!isReturning) {

if (\_i < 16) {

setObjectY(getObjectY() - 4);

}

\_i++;

} else {

if (\_i < 2)

setObjectY(getObjectY() - 4);

else if (\_i < 4)

setObjectY(getObjectY() + 4);

if (++\_i == 4) {

\_i = 16;

}

}

}

/\*\*

\* moveDown animation uses \_i as progress does diffrent animations based on isReturning

\*

\* @param isReturning

\* - if true the animation will be a bump else the animation would be a block

\* movement

\*/

public void moveDown(boolean isReturning) {

if (!isReturning) {

if (\_i < 16) {

setObjectY(getObjectY() + 4);

}

\_i++;

} else {

if (\_i < 2)

setObjectY(getObjectY() + 4);

else if (\_i < 4)

setObjectY(getObjectY() - 4);

if (++\_i == 4) {

\_i = 16;

}

}

}

/\*\*

\* Attack function of the movingUnit ( ranged or melee)

\*/

public abstract void attackRight();

/\*\*

\* Attack function of the movingUnit ( ranged or melee)

\*/

public abstract void attackLeft();

/\*\*

\* Attack function of the movingUnit ( ranged or melee)

\*/

public abstract void attackUp();

/\*\*

\* Attack function of the movingUnit ( ranged or melee)

\*/

public abstract void attackDown();

/\*\*

\* get the no padding X coordinate

\*

\* @return the X position ( in pixels)

\*/

public abstract int getRealX();

/\*\*

\* get the no padding Y coordinate

\*

\* @return the Y position (in pixels)

\*/

public abstract int getRealY();

/\*\*

\* get the no padding height

\*

\* @return - the height of the Panel without padding(in pixels)

\*/

public abstract int getRealHeight();

/\*\*

\* get the no padding width

\*

\* @return - the width of the Panel without padding(in pixels)

\*/

public abstract int getRealWidth();

/\*\*

\* Check if a certain place is blocked by other movingUnits or walls

\*

\* @param i

\* - the I coordinate ( in position units)

\* @param j

\* - the J coordinate ( in position units)

\* @return

\*/

public boolean isBlocked(int i, int j) {

boolean flag = (MapG.get\_map()[i][j].getWallLevel() != 0);

flag = flag || GameStats.occupiedExists(i, j);

return flag;

}

/\*\*

\* remove the moving unit from the Map and remove the global Unit List, stop all timers and

\* nullify the graphic class

\*/

public void removeFromMap() {

// \_timer.removeActionListener(this);

MapGPanel.getInstance().remove(this);

GameStats.getPlayers().remove(this);

\_timer.stop();

try {

\_fireTimer.stop();

} catch (NullPointerException e) {

}

;

setObjectGraphicClass(null);

}

}

**unit.ObjectOnMap**

package unit;

import java.awt.Color;

import java.awt.Graphics;

import javax.swing.JPanel;

import map.MapG;

import map.MapGPanel;

import java.awt.Rectangle;

import ultilityTools.PaintingInterface;

/\*\*

\* ObjectOnMap is a class that represents an object on map that is not part of it.

\*

\* @author Peter

\*/

public class ObjectOnMap extends JPanel {

protected int \_x;

protected int \_y;

protected int \_width;

protected int \_height;

/\*\* the graphic of the Object \*/

protected PaintingInterface \_graphicClass = null;

/\*\*

\* Constructor

\*

\* @param x

\* - the x position

\* @param y

\* - the y position

\* @param width

\* - the width of the object

\* @param height

\* - the height of the object

\*/

public ObjectOnMap(int x, int y, int width, int height) {

setOpaque(false);

setBoundForObject(x, y, width, height);

}

/\*\*

\* Set bounds for the object

\*

\* @param x

\* - the x position

\* @param y

\* - the y position

\* @param width

\* - the width of the object

\* @param height

\* - the height of the object

\*/

public void setBoundForObject(int x, int y, int width, int height) {

\_x = x;

\_y = y;

\_width = width;

\_height = height;

setBounds(\_x, \_y, \_width, \_height);

}

/\*\*

\* the painting method , if the object wants to be drawn it must implement paintingInterface

\* @param g - the graphics used

\*/

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

// System.out.println("Painting");

if (\_graphicClass != null)

if (\_graphicClass instanceof PaintingInterface) {

//MapGPanel.getInstance().repaint();

\_graphicClass.myPaintComponent(g);

}

}

/\*\*

\* when the center of one panel is on the border of the other panel

\*

\* @param collideObj the object to check

\* @return true if the center of this panel is within the other panel

\*/

public boolean isColliding(ObjectOnMap collideObj) {

return (isCollidingHorizontal(collideObj) && isCollidingVertical(collideObj));

}

/\*\*

\* check if the current object touches the checked object from the left or right

\* @param collideObj - the checked object

\* @return true if they collide else false

\*/

public boolean isCollidingHorizontal(ObjectOnMap collideObj) {

return (isCollidingLeft(collideObj) && isCollidingRight(collideObj));

}

/\*\*

\* check if the current object touches the checked object from the top or bottom

\* @param collideObj - the checked object

\* @return true if they collide else false

\*/

public boolean isCollidingVertical(ObjectOnMap collideObj) {

return (isCollidingUp(collideObj) && isCollidingDown(collideObj));

}

/\*\*

\* check if the current object touches the checked object from the left

\* @param collideObj - the checked object

\* @return true if they collide else false

\*/

public boolean isCollidingRight(ObjectOnMap collideObj) {

int maxX = collideObj.getObjectX() + collideObj.getObjectWidth() - \_width / 2;

return (\_x < maxX && isTouching(collideObj));

}

/\*\*

\* check if the current object touches the checked object from the right

\* @param collideObj - the checked object

\* @return true if they collide else false

\*/

public boolean isCollidingLeft(ObjectOnMap collideObj) {

int minX = collideObj.getObjectX() - \_width / 2;

return (\_x > minX && isTouching(collideObj));

}

/\*\*

\* check if the current object touches the checked object from the bottom

\* @param collideObj - the checked object

\* @return true if they collide else false

\*/

public boolean isCollidingUp(ObjectOnMap collideObj) {

int minY = collideObj.getObjectY() - \_height / 2;

return (\_y > minY && isTouching(collideObj));

}

/\*\*

\* check if the current object touches the checked object from the top

\* @param collideObj - the checked object

\* @return true if they collide else false

\*/

public boolean isCollidingDown(ObjectOnMap collideObj) {

int maxY = collideObj.getObjectY() + collideObj.getObjectHeight() - \_height / 2;

return (\_y < maxY && isTouching(collideObj));

}

/\*\*

\* check if borders collide between an object

\* @param collideObj - the checked object

\* @return true if they collide else false

\*/

public boolean isTouching(ObjectOnMap collideObj) {

Rectangle r = new Rectangle(\_x, \_y, \_width, \_height);

Rectangle t = new Rectangle(collideObj.getObjectX(), collideObj.getObjectY(), collideObj.getObjectWidth(),

collideObj.getObjectHeight());

return r.intersects(t);

}

/\*

\* every set uses setBoundForObject

\*/

public int getObjectX() {

return \_x;

}

public void setObjectX(int x) {

setBoundForObject(x, \_y, \_width, \_height);

}

public int getObjectY() {

return \_y;

}

public void setObjectY(int y) {

setBoundForObject(\_x, y, \_width, \_height);

}

public int getObjectWidth() {

return \_width;

}

public void setObjectWidth(int width) {

setBoundForObject(\_x, \_y, width, \_height);

}

public int getObjectHeight() {

return \_height;

}

public void setObjectHeight(int height) {

setBoundForObject(\_x, \_y, \_width, height);

}

public void setObjectGraphicClass(PaintingInterface graphicClass) {

\_graphicClass = graphicClass;

}

}

**unit.Projectile**

package unit;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.LinkedList;

import javax.swing.JPanel;

import javax.swing.Timer;

import engine.Block;

import images.Img;

import mainInitialize.GameStats;

import mainInitialize.Main;

import map.MapG;

import map.MapGPanel;

import ultilityTools.PaintingInterface;

import unit.enemy.Enemy;

/\*\*

\* Projectile is a class that represents a bullet (currently only for player)

\*

\* @author Peter

\*/

public class Projectile extends ObjectOnMap implements ActionListener {

private int \_speed;

private LinkedList<Block> \_colidedBlocks;

private LinkedList<ObjectOnMap> \_colidedUnits;

private double \_angle;

private double \_directionAngle;

private Timer \_timer;

private int \_bulletHp;

private int \_bulletDamage;

private ProjectileG \_graphics;

private int \_xMove;

private int \_yMove;

/\*\*

\* Constructor

\*

\* @param x

\* - the starting X position (in Pixels)

\* @param y

\* - the starting Y position (in Pixels)

\* @param width

\* - the width of the projectile panel

\* @param height

\* - the height of the projectile panel

\* @param speed

\* - the speed of the projectile

\* @param angle

\* - the angle at which to turn the projectile to

\* @param directionAngle

\* - the angle of cone (currently only direct projectiles)

\*/

public Projectile(int x, int y, int width, int height, int speed, double angle, double directionAngle) {

super(x, y, width, height);

\_xMove = \_yMove = 0;

MapGPanel.getInstance().add(this);// dont forget to add to panel after creating new panel\

\_colidedBlocks = new LinkedList<Block>();

\_colidedUnits = new LinkedList<ObjectOnMap>();

// setBackground(Color.gray);

\_bulletHp = 50;

\_bulletDamage = 30;

\_speed = speed;

\_timer = new Timer(97, this);

\_angle = angle;

\_directionAngle = directionAngle;

calculateNewCordinates();

\_graphics = new ProjectileG(getObjectWidth(), getObjectHeight(), \_angle, \_directionAngle);

setObjectGraphicClass(\_graphics);

// System.out.println(getObjectWidth() + " " + getObjectHeight());

\_timer.start();

}

/\*\*

\* insert the appropriate pixels moved and the height \ width of the Panel to fit the rotated

\* projectile

\*/

public void calculateNewCordinates() {

double sin = Math.abs(Math.sin(Math.toRadians(\_angle)));

double cos = Math.abs(Math.cos(Math.toRadians(\_angle)));

int neww = (int) Math.floor(getObjectWidth() \* cos + getObjectHeight() \* sin);

int newh = (int) Math.floor(getObjectHeight() \* cos + getObjectWidth() \* sin);

setObjectWidth(neww);

setObjectHeight(newh);

\_xMove = (int) (Math.sin(Math.toRadians(\_directionAngle)) \* \_speed);

\_yMove = (int) (Math.cos(Math.toRadians(\_directionAngle)) \* \_speed);

// System.out.println(\_xMove + "" + \_yMove);

}

/\*\*

\* is called whenever a timer tick occours, does an action collides with an object ,self

\* destructs if health reaches 0

\*/

@Override

public void actionPerformed(ActionEvent e) {

setObjectX(getObjectX() + \_xMove);

setObjectY(getObjectY() + \_yMove);

\_graphics.copyProjectile(getObjectWidth(), getObjectHeight(), \_angle, \_directionAngle);

// \_bulletHp-=1;

actionIfCollision();

// repaint();

if (\_bulletHp <= 0)

killProjectile();

}

/\*\*

\* the function checks if collides in one of the directions

\* @param collideObj -the checked object

\* @return true if both objects collide

\*/

public boolean checkProperCollisions(ObjectOnMap collideObj) {

return ((\_xMove > 0 && isCollidingLeft(collideObj)) || (\_xMove < 0 && isCollidingRight(collideObj))

|| (\_yMove > 0 && isCollidingUp(collideObj)) || (\_yMove < 0 && isCollidingDown(collideObj)));

}

/\*\*

\* do an action is struck by wall , halfwall or Moving unit

\*/

public void actionIfCollision() {// problem is in here

ObjectOnMap tempObj;

for (Block temp : GameStats.getFullWalls()) {

tempObj = new ObjectOnMap(temp.getX() \* temp.getWidth(), temp.getY() \* temp.getHeight(), temp.getWidth(),

temp.getHeight());

if (checkProperCollisions(tempObj)

&& (\_colidedBlocks.isEmpty() || !\_colidedBlocks.getLast().equals(temp))) {

\_colidedBlocks.add(temp);

\_bulletHp = 0;

return;

}

}

for (Block temp : GameStats.getHalfWalls()) {

tempObj = new ObjectOnMap(temp.getX() \* temp.getWidth(), temp.getY() \* temp.getHeight(), temp.getWidth(),

temp.getHeight());

if (checkProperCollisions(tempObj)

&& (\_colidedBlocks.isEmpty() || !\_colidedBlocks.getLast().equals(temp))) {

\_colidedBlocks.add(temp);

\_bulletHp -= temp.getDefenseVal();

temp.setCurrentHealth(temp.getCurrentHealth() - \_bulletDamage);

return;

}

}

Enemy enemyTemp = null;

for (MovingUnit temp : GameStats.getPlayers()) {

if (temp instanceof Enemy && (\_colidedUnits.isEmpty() || !\_colidedUnits.getLast().equals(temp))) {

enemyTemp = (Enemy) temp;

tempObj = new ObjectOnMap(enemyTemp.getRealX(), enemyTemp.getRealY(), enemyTemp.getRealWidth(),

enemyTemp.getRealHeight());

if (checkProperCollisions(tempObj)) {

\_colidedUnits.add(temp);

enemyTemp.getStats().setHealth(enemyTemp.getStats().getHealth() - \_bulletDamage);

System.out.println("Enemy HP : " + enemyTemp.getStats().getHealth());

return;

}

}

}

}

/\*\*

\* destroy the projectile and remove from the map panel

\*/

public void killProjectile() {

setObjectGraphicClass(null);

repaint();

\_timer.stop();

\_timer.removeActionListener(this);

MapGPanel.getInstance().remove(this);

}

}

**unit.ProjectileG**

package unit;

import images.Img;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.Image;

import javax.swing.ImageIcon;

import ultilityTools.ImageTools;

import ultilityTools.PaintingInterface;

/\*\*

\* ProjectileG is a class that represents the graphic class of a projectile

\*

\* @author Peter

\*/

public class ProjectileG implements PaintingInterface {

private Img \_projectileImg;

private int \_width;

private int \_height;

private double \_angle;

private double \_directionAngle;

/\*\*

\* Constructor

\*

\* @param width

\* - the width of the projectile (in pixels)

\* @param height

\* - the height of the projectile (in pixels)

\* @param angle

\* - the angle that indicates to which direction to turn the projectile

\* @param directionAngle

\* - the angle of the cone

\*/

public ProjectileG(int width, int height, double angle, double directionAngle) {

copyProjectile(width, height, angle, directionAngle);

\_projectileImg = new Img("Pawns\\Weapons\\Animations\\Bullet.png", 0, 0, \_width, \_height);

\_projectileImg.setImage(ImageTools.rotate(\_projectileImg, \_angle));

// \_projectileImg.setImage(ImageTools.rotate(\_projectileImg, \_angle));

}

/\*\*

\* sets the variables with the received parameters

\*

\* @param width

\* - the width of the projectile (in pixels)

\* @param height

\* - the height of the projectile (in pixels)

\* @param angle

\* - the angle that indicates to which direction to turn the projectile

\* @param directionAngle

\* - the angle of the cone

\*/

public void copyProjectile(int width, int height, double angle, double directionAngle) {

\_width = width;

\_height = height;

\_angle = angle;

\_directionAngle = directionAngle;

}

/\*\*

\* the PaintComponent of the img

\*/

@Override

public void myPaintComponent(Graphics g) {

// \_projectileImg.drawImgRotate(g, \_angle);

// g.setColor(Color.blue);

// g.fillRect(0, 0, \_width, \_height);

\_projectileImg.drawImg(g);

}

}

**unit.WinningInterface**

package unit;

/\*\*

\* Interface of a winning unit (Player\Enemy) all units that play must implement.

\* @author Peter

\*

\*/

public interface WinningInterface {

public void pauseUnit();

public void startUnit();

//public void removeUnit();

public void actionIfWin();

public boolean isWin();

//public int getRealX();

//public int getRealY();

}

**unit.customizeScreen.CustomizeFrame**

package unit.customizeScreen;

import javax.swing.JFrame;

import unit.player.PlayerG;

/\*\*

\* CustomizeFrame is a class that represents a frame with Player custom options

\*

\* @author Peter

\* @category Singleton - only contains one instance

\*/

public class CustomizeFrame extends JFrame {

private static CustomizeFrame singleton = null;

/\*\*

\* Constructor inits the Customize panel and shows it to the user

\*/

private CustomizeFrame() {

add(CustomizeScreen.init());

setVisible(true);

setSize(600, 600);

}

/\*\*

\* initialize the singleton instance

\*

\* @return CustomizeFrame only instance

\*/

public static CustomizeFrame init() {

if (singleton == null)

singleton = new CustomizeFrame();

return singleton;

}

/\*\*

\* get the Singleton instance

\*

\* @return CustomizeFrame only instance

\*/

public static CustomizeFrame getInstance() {

return singleton;

}

}

**unit.customizeScreen.CustomizeInterface**

package unit.customizeScreen;

/\*\*

\* CustomizeInterface is an interface implemented by PlayerG in order to notify when a Clothing button is clicked

\* @author Peter

\*

\*/

public interface CustomizeInterface {

void CustomizeButtonClicked(String name,boolean isNext);

}

**unit.customizeScreen.CustomizeScreen**

package unit.customizeScreen;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.LinkedList;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JLabel;

import javax.swing.JPanel;

import mainInitialize.GameStats;

import mainInitialize.MainMenu;

import unit.player.Player;

import unit.player.PlayerG;

/\*\*

\* CustomizeScreen represents a Panel on the CustomizeFrame Contains buttons and category of weapons

\*

\* @author Peter

\*/

public class CustomizeScreen extends JPanel {

private static CustomizeScreen singleton = null;

private String \_nameList[] = { "Headgear", "Hair", "Face", "Body", "Top", "Weapon", "Bottom" };

private String \_weaponTypes[] = { "Energy", "Heavy", "Melee", "Pistols", "Rifles", "Shotguns", "SMGs" };

private LinkedList<OptimizingButton> \_nextButtons;

private JPanel \_nextPanel;

private LinkedList<OptimizingButton> \_prevButtons;

private JPanel \_prevPanel;

private JPanel \_bottomPanel;

private JButton \_backButton;

private JComboBox<String> \_weaponSelector;

/\*\*

\* Constructor prepares all buttons and appearal sets

\*/

private CustomizeScreen() {

DisplayScreen.init();

// setBackground(Color.black);

\_weaponSelector = new JComboBox<String>(\_weaponTypes);

\_weaponSelector.setSelectedIndex(4);

\_weaponSelector.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

DisplayScreen.getInstance().getPlayer().getCustomize().fillAllApparels();

DisplayScreen.getInstance().getPlayer().getCustomize().getWeapon().resetCurrent();

DisplayScreen.getInstance().getPlayer().setImgSide(0);

DisplayScreen.getInstance().getPlayer().setAllSizes(DisplayScreen.\_height, DisplayScreen.\_width);

DisplayScreen.getInstance().repaint();

}

});

\_nextPanel = new JPanel();

\_prevPanel = new JPanel();

\_bottomPanel = new JPanel();

\_nextPanel.setOpaque(false);

\_prevPanel.setOpaque(false);

\_bottomPanel.setOpaque(false);

// setOpaque(false);

// \_displayUnitScreen.setBackground(Color.black);

setBackground(Color.pink);

\_nextButtons = new LinkedList<OptimizingButton>();

\_prevButtons = new LinkedList<OptimizingButton>();

setLayout(new BorderLayout());

\_nextPanel.setLayout(new GridLayout(8, 1, 10, 10));

for (int i = 0; i < \_nameList.length; i++) {

\_nextButtons.add(new OptimizingButton(\_nameList[i], true));

setButtonListener(\_nextButtons.get(i));

\_nextPanel.add(\_nextButtons.get(i).getBtn());

}

\_prevPanel.setLayout(new GridLayout(8, 1, 10, 10));

for (int i = 0; i < \_nameList.length; i++) {

\_prevButtons.add(new OptimizingButton(\_nameList[i], false));

setButtonListener(\_prevButtons.get(i));

\_prevPanel.add(\_prevButtons.get(i).getBtn());

}

\_bottomPanel.setLayout(new GridLayout(2, 2));

\_bottomPanel.add(new JLabel("WEAPON:"));

\_bottomPanel.add(\_weaponSelector);

\_backButton = new JButton("BACK");

\_backButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if (MainMenu.getInstance() != null)

MainMenu.getInstance().setVisible(true);

CustomizeFrame.getInstance().setVisible(false);

}

});

\_bottomPanel.add(\_backButton);

add(\_nextPanel, BorderLayout.EAST);

add(\_prevPanel, BorderLayout.WEST);

add(\_bottomPanel, BorderLayout.SOUTH);

add(DisplayScreen.getInstance(), BorderLayout.CENTER);

}

/\*\*

\* initialize the singleton instance

\*

\* @return CustomizeScreen only instance

\*/

public static CustomizeScreen init() {

if (singleton == null)

singleton = new CustomizeScreen();

return singleton;

}

/\*\*

\* get the Singleton instance

\*

\* @return CustomizeScreen only instance

\*/

public static CustomizeScreen getInstance() {

return singleton;

}

/\*\*

\* sets the button to listener for clicks

\*

\* @param optBtn

\* -the button we want to set as listener

\*/

public void setButtonListener(OptimizingButton optBtn) {

optBtn.getBtn().addActionListener(optBtn);

}

/\*\*

\* @return the name of the weapon type selected

\*/

public String getWeaponSelected() {

return \_weaponTypes[\_weaponSelector.getSelectedIndex()];

}

/\*\*

\* @param args

\*/

/\*

\* public static void main(String[] args) { CustomizeFrame frame = CustomizeFrame.init(); }

\*/

}

**unit.customizeScreen.DisplayScreen**

package unit.customizeScreen;

import java.awt.Graphics;

import javax.swing.JPanel;

import mainInitialize.GameStats;

import unit.player.Player;

import unit.player.PlayerG;

/\*\*

\* The panel that contains the visuals of the player Custom

\*

\* @author Peter

\* @category Singleton - only contains one instance

\*/

public class DisplayScreen extends JPanel {

// static playerG

private static PlayerG \_player;

private static DisplayScreen singleton = null;

public static int \_width;

public static int \_height;

/\*\*

\* constructor initates the PlayerG

\*/

private DisplayScreen() {

\_width = 400;

\_height = 550;

\_player = new PlayerG(\_height, \_width);

// \_player.setParent(this);

}

/\*\*

\* initialize the singleton instance

\*

\* @return DisplayScreen only instance

\*/

public static DisplayScreen init() {

if (singleton == null)

singleton = new DisplayScreen();

return singleton;

}

/\*\*

\* get the Singleton instance

\*

\* @return DisplayScreen only instance

\*/

public static DisplayScreen getInstance() {

return singleton;

}

/\*\*

\* copy the charachter from game ( if exists) to the display screen

\*/

public static void copyToDisplay() {

if (GameStats.getInstance().getPlayers().size() != 0)

\_player.copyPlayerG(((Player) GameStats.getInstance().getPlayers().get(0)).getPlayerGraphics());

\_player.setParent(singleton);

\_player.setAllSizes(\_height, \_width);

}

/\*\*

\* The paint component calls the Paint Component of the PlayerG

\*/

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

\_player.myPaintComponent(g);

}

/\*\*

\* @return the graphics for the player

\*/

public PlayerG getPlayer() {

return \_player;

}

}

**unit.customizeScreen.OptimizingButton**

package unit.customizeScreen;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

import unit.player.PlayerG;

/\*\*

\* OptimizingButton is a class containing a button and a listener ( PlayerG) that gets alerted

\* whenever a button the button is pressed

\*

\* @author Peter

\*/

public class OptimizingButton implements ActionListener {

private JButton \_btn;

private String \_name;

private boolean \_isNext;

private PlayerG \_listener;

/\*\*

\* Constructor

\* @param name - the label of the button

\* @param isNext - is it a reverse button ( next or back)

\*/

public OptimizingButton(String name, boolean isNext) {

\_listener = DisplayScreen.getInstance().getPlayer();

\_isNext = isNext;

\_name = new String(name);

\_btn = new JButton(\_name);

}

/\*\*

\* Gets called whenever the button is clicked, notifies the Listener about the click

\*/

@Override

public void actionPerformed(ActionEvent e) {

if (\_isNext) {

System.out.println("Item Next : " + \_name);

} else {

System.out.println("Item Prev : " + \_name);

}

\_listener.CustomizeButtonClicked(\_name, \_isNext);

}

/\*\*

\* @return the label of the button

\*/

public String getName() {

return \_name;

}

/\*\*

\* set a new Label for the button

\* @param name - the new label for the button

\*/

public void setName(String name) {

\_name = name;

}

/\*\*

\* @return the button in this class

\*/

public JButton getBtn() {

return \_btn;

}

}

**unit.enemy.Enemy**

package unit.enemy;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.util.LinkedList;

import java.util.Random;

import javax.swing.JDialog;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.Timer;

import engine.Unit;

import engine.dijekstra.SmartMove;

import engine.input.KeyboardInput;

import mainInitialize.GameStats;

import map.MapG;

import map.MapGFrame;

import ultilityTools.PaintingInterface;

import unit.FireCooldown;

import unit.MovingUnit;

import unit.WinningInterface;

import unit.ObjectOnMap;

import unit.Projectile;

import unit.player.Player;

import unit.player.PlayerG;

/\*\*

\* The Enemy class is a class that represents the enemy panel

\*

\* @author Peter

\*/

public class Enemy extends MovingUnit implements ActionListener, WinningInterface {

/\*\* the stats of the enemy \*/

private Unit \_stats;

/\*\* the graphic class of the enemy \*/

private EnemyG \_graphics;

/\*\* if the enemy is busy doing one of the actions the flag will be active \*/

private boolean \_rightBusy = false;

private boolean \_leftBusy = false;

private boolean \_upBusy = false;

private boolean \_downBusy = false;

private boolean \_busy = false;

private boolean \_isAttacking = false;

private boolean \_isBlocked;

private int \_opt;

private int \_damageToWall = 20;

private SmartMove \_mover;

private Unit \_hunted;

/\*\*

\* Constructor

\*

\* @param x

\* - the graphical X position of the enemy

\* @param y

\* - the graphical Y position of the enemy

\* @param width

\* - the width of the enemy

\* @param height

\* - the height of the enemy

\* @param map

\* - the Graphical map

\*/

public Enemy(int x, int y, int width, int height) {

/\*\* use Object's on map constructor \*/

super(x, y, width, height);

/\*\* initialize stats \*/

\_stats = new Unit('E', x / 64, y / 64, 500, this);

/\*\* create new Graphics \*/

\_graphics = new EnemyG(\_height, \_width);

setObjectGraphicClass(\_graphics);

/\*\* the action timer \*/

\_timer = new Timer(28, this);// 40

\_timer.start();

\_mover = new SmartMove();

\_hunted = ((Player) (GameStats.getPlayers().get(0))).getStats();

}

/\*\*

\* The action that is done after every tick checks if the enemy won , checks if he is busy if

\* not check if he needs to attack

\*/

@Override

public void actionPerformed(ActionEvent e) {

actionIfWin();

// System.out.println("Hunted Coordinates: "+ \_hunted.getCurrentX() +" , " +

// \_hunted.getCurrentY());

if (!\_leftBusy && !\_rightBusy && !\_upBusy && !\_downBusy)

\_opt = \_mover.getSmartMove(\_stats, \_hunted.getCurrentX(), \_hunted.getCurrentY());

/\*\*

\* if 'A' is held and none of the actions are done or the left action is currently busy

\*/

if ((\_opt == 3 && !\_busy) || (\_leftBusy && \_busy)) {

\_leftBusy = true;

\_busy = true;

if (\_i == 0) {

\_isBlocked = isBlocked(\_stats.getCurrentX() - 1, \_stats.getCurrentY());

/\*\* if the location player moves is floor \*/

if (!\_isBlocked)

\_stats.setCurrentX(\_stats.getCurrentX() - 1);

GameStats.addOccupied(\_stats.getCurrentX(), \_stats.getCurrentY());

}

/\*\* Move left ,receives boolean that tells which animation cycle to do \*/

if (\_isBlocked)

if (!\_isAttacking)

attackLeft();

else

moveLeft(true);

else

moveLeft(false);

/\*\* set direction in order to draw Properly \*/

\_graphics.setImgSide(3);

/\*\* after finishing cycle \*/

if (\_i == 16) {

if (!\_isBlocked)

GameStats.removeOccupied(\_stats.getCurrentX() + 1, \_stats.getCurrentY());

/\*\* reset the \_i and free the busy flags \*/

\_i = 0;

\_leftBusy = false;

\_busy = false;

\_isBlocked = false;

}

// repaint();

return;

}

/\*\*

\* if 'D' is held and none of the actions are done or the right action is currently busy

\*/

if (((\_opt == 1 && !\_busy)) || (\_rightBusy && \_busy)) {

\_rightBusy = true;

\_busy = true;

if (\_i == 0) {

\_isBlocked = isBlocked(\_stats.getCurrentX() + 1, \_stats.getCurrentY());

/\*\* if the location player moves is floor \*/

if (!\_isBlocked)

\_stats.setCurrentX(\_stats.getCurrentX() + 1);

GameStats.addOccupied(\_stats.getCurrentX(), \_stats.getCurrentY());

}

/\*\* Move right ,receives boolean that tells which animation cycle to do \*/

if (\_isBlocked)

if (!\_isAttacking)

attackRight();

else

moveRight(true);

else

moveRight(false);

/\*\* set direction in order to draw Properly \*/

\_graphics.setImgSide(1);

/\*\* after finishing cycle \*/

if (\_i == 16) {

if (!\_isBlocked)

GameStats.removeOccupied(\_stats.getCurrentX() - 1, \_stats.getCurrentY());

/\*\* reset the \_i and free the busy flags \*/

\_i = 0;

\_rightBusy = false;

\_busy = false;

\_isBlocked = false;

}

// repaint();

return;

}

/\*\*

\* if 'W' is held and none of the actions are done or the up action is currently busy

\*/

if (((\_opt == 2 && !\_busy)) || (\_upBusy && \_busy)) {

\_upBusy = true;

\_busy = true;

/\*\* Move up , receives boolean that tells which animation cycle to do \*/

if (\_i == 0) {

\_isBlocked = isBlocked(\_stats.getCurrentX(), \_stats.getCurrentY() - 1);

/\*\* if the location player moves is floor \*/

if (!\_isBlocked)

\_stats.setCurrentY(\_stats.getCurrentY() - 1);

GameStats.addOccupied(\_stats.getCurrentX(), \_stats.getCurrentY());

}

if (\_isBlocked)// MapG.get\_map()[\_stats.getCurrentX()][\_stats.getCurrentY()

// - 1].getWallLevel()

// != 0)

/\*\* if not in middle of attack \*/

if (!\_isAttacking)

attackUp();

else

moveUp(true);

else

moveUp(false);

/\*\* set direction in order to draw Properly \*/

\_graphics.setImgSide(2);

/\*\* after finishing cycle \*/

if (\_i == 16) {

if (!\_isBlocked)

GameStats.removeOccupied(\_stats.getCurrentX(), \_stats.getCurrentY() + 1);

/\*\* reset the \_i and free the busy flags \*/

\_i = 0;

\_upBusy = false;

\_busy = false;

\_isBlocked = false;

}

// repaint();

return;

}

/\*\*

\* if 'S' is held and none of the actions are done or the down action is currently busy

\*/

if (((\_opt == 0 && !\_busy)) || (\_downBusy && \_busy)) {

\_downBusy = true;

\_busy = true;

if (\_i == 0) {

\_isBlocked = isBlocked(\_stats.getCurrentX(), \_stats.getCurrentY() + 1);

if (!\_isBlocked)

/\*\* if the location player moves is floor \*/

\_stats.setCurrentY(\_stats.getCurrentY() + 1);

GameStats.addOccupied(\_stats.getCurrentX(), \_stats.getCurrentY());

}

/\*\* Move down , receives boolean that tells which animation cycle to do \*/

if (\_isBlocked)// MapG.get\_map()[\_stats.getCurrentX()][\_stats.getCurrentY()

// + 1].getWallLevel()

// != 0)

if (!\_isAttacking)

attackDown();

else

moveDown(true);

else

moveDown(false);

/\*\* set direction in order to draw Properly \*/

\_graphics.setImgSide(0);

/\*\* after finishing cycle \*/

if (\_i == 16) {

if (!\_isBlocked)

GameStats.removeOccupied(\_stats.getCurrentX(), \_stats.getCurrentY() - 1);

/\*\* reset the \_i and free the busy flags \*/

\_i = 0;

\_downBusy = false;

\_busy = false;

\_isBlocked = false;

}

// repaint();

return;

}

if (\_opt != 0) {

\_graphics.setImgSide(0);

repaint();

}

}

/\*\*

\* set the new isAttacking flag which meansif the enemy is attacking at the moment

\*

\* @param isAttacking

\* - the new is Attacking flag of the enemy

\*/

public void setAttacking(boolean isAttacking) {

\_isAttacking = isAttacking;

}

/\*\*

\* @return the damage a single attack of the enemy does to a wall

\*/

public int getDamageToWall() {

return \_damageToWall;

}

/\*\*

\* notifies the user if the enemy has won

\*/

public void actionIfWin() {

if (isWin()) {

JDialog jd;

jd = new JDialog(MapGFrame.getInstance(), "Test Dialog");

JPanel winPanel = new JPanel();

winPanel.add(new JLabel("Enemy Won!"));

jd.add(winPanel);

jd.pack();

jd.setVisible(true);

GameStats.pauseGame();

}

}

/\*\*

\* checks if there are no players left if true the enemy has won .

\*/

public boolean isWin() {

for (MovingUnit temp : GameStats.getPlayers()) {

if (temp instanceof Player)

return false;

}

return true;// (getObjectX() == (7 \* 64) && getObjectY() == (7 \* 64));

}

/\*\*

\* pauses the unit , used for pausing the game

\*/

public void pauseUnit() {

\_timer.stop();

if (\_fireTimer != null)

\_fireTimer.stop();// enemy doesnt use fire timer now .

}

/\*\*

\* starts the unit , used for continueing the game

\*/

public void startUnit() {

\_timer.start();

// \_fireTimer.start();

}

/\*\*

\* start to attack the right direction

\*/

@Override

public void attackRight() {

new EnemyAttack(getObjectX() + 64, getObjectY(), 64, 64, \_damageToWall, 90, this);

}

/\*\*

\* start to attack the left direction

\*/

@Override

public void attackLeft() {

new EnemyAttack(getObjectX() - 64, getObjectY(), 64, 64, \_damageToWall, 270, this);

}

/\*\*

\* start to attack the upper direction

\*/

@Override

public void attackUp() {

new EnemyAttack(getObjectX(), getObjectY() - 64, 64, 64, \_damageToWall, 180, this);

}

/\*\*

\* start to attack the bottom direction

\*/

@Override

public void attackDown() {

new EnemyAttack(getObjectX(), getObjectY() + 64, 64, 64, \_damageToWall, 180, this);

}

/\*\*

\* @returns the ObjectX without the padding ( in pixels)

\*/

@Override

public int getRealX() {

return getObjectX();

}

/\*\*

\* @returns the ObjectY without the padding ( in pixels)

\*/

@Override

public int getRealY() {

return getObjectY();

}

/\*\*

\* @returns the Object Height without the padding ( in pixels)

\*/

@Override

public int getRealHeight() {

return getObjectHeight();

}

/\*\*

\* @returns the Object Width without the padding ( in pixels)

\*/

@Override

public int getRealWidth() {

return getObjectWidth();

}

/\*\*

\* @return the Enemy stats

\*/

public Unit getStats() {

return \_stats;

}

}

**unit.enemy.EnemyAttack**

package unit.enemy;

import java.awt.Color;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.Timer;

import engine.Block;

import engine.Unit;

import mainInitialize.GameStats;

import map.MapGPanel;

import unit.MovingUnit;

import unit.ObjectOnMap;

import unit.ProjectileG;

import unit.player.Player;

import unit.player.PlayerG;

/\*\*

\* EnemyAttack is a class that represents the melee attack of an enemy

\*

\* @author Peter

\*/

public class EnemyAttack extends ObjectOnMap implements ActionListener, MeleeAttackInterface {

private Timer \_timer;

private EnemyAttackG \_graphics;

private int \_attackDamage;

private double \_directionAngle;

private Enemy \_parent;

/\*\*

\* Constructor

\*

\* @param x

\* - the starting X position (in Pixels)

\* @param y

\* - the starting Y position (in Pixels)

\* @param width

\* - the width of the Enemy Attack panel (in Pixels)

\* @param height

\* - the height of the Enemy Attack panel (in Pixels)

\* @param attackDamage

\* - the attack damage of the attack

\* @param directionAngle

\* - the direction of the attack in degrees

\* @param parent

\* - the parent Panel of the attack

\*/

public EnemyAttack(int x, int y, int width, int height, int attackDamage, double directionAngle, Enemy parent) {

super(x, y, width, height);

\_parent = parent;

\_parent.setAttacking(true);

MapGPanel.getInstance().add(this);

\_attackDamage = attackDamage;

\_directionAngle = directionAngle;

\_timer = new Timer(200, this);// 200

\_graphics = new EnemyAttackG(getObjectWidth(), getObjectHeight(), \_directionAngle, this);

setObjectGraphicClass(\_graphics);

\_timer.start();

}

/\*\*

\* activates whenever a timer ticks

\*/

@Override

public void actionPerformed(ActionEvent e) {

actionIfHalfWall();

actionIfPlayer();

\_graphics.nextAttackImg();

repaint();

}

/\*\*

\* do action if collides with halfwall

\*/

public void actionIfHalfWall() {

for (Block temp : GameStats.getHalfWalls()) {

ObjectOnMap tempObj = new ObjectOnMap(temp.getX() \* temp.getWidth(), temp.getY() \* temp.getHeight(),

temp.getWidth(), temp.getHeight());

if (isColliding(tempObj)) {

// System.out.println("I am eating" +temp.getX()+":"+temp.getY());

temp.setCurrentHealth(temp.getCurrentHealth() - \_attackDamage);

/\*

\* System.out.println("Hp left " + temp.getCurrentHealth());

\* System.out.println(\_bulletHp);

\*/

return;

}

}

}

/\*\*

\* do action if collides with player

\*/

public void actionIfPlayer() {

for (MovingUnit temp : GameStats.getPlayers()) {

if (temp instanceof Player) {

Unit tempPlayerStats = ((Player) temp).getStats();

ObjectOnMap tempObj = new ObjectOnMap(temp.getRealX(), temp.getRealY(), temp.getRealWidth(),

temp.getRealHeight());

if (isColliding(tempObj)) {

tempPlayerStats.setHealth(tempPlayerStats.getHealth() - \_attackDamage);

// System.out.println("I am eating player at" + ((Player)temp).getRealX() + " :

// " + ((Player)temp).getRealY());

// System.out.println("Player HP:" + ((Player) temp).getStats().getHealth());

}

}

}

}

/\*\*

\* removes the enemy attack from the map when notified about animation ending.

\*/

@Override

public void meleeAttackFinished() {

setObjectGraphicClass(null);

repaint();

\_timer.stop();

\_timer.removeActionListener(this);

MapGPanel.getInstance().remove(this);

\_parent.setAttacking(false);

\_parent = null;

}

}

**unit.enemy.EnemyAttackG**

package unit.enemy;

import java.awt.Graphics;

import images.Img;

import ultilityTools.ImageTools;

import ultilityTools.PaintingInterface;

/\*\*

\* EnemyAttackG is a class that represents the graphics of an Enemy attack

\* @author Peter

\*

\*/

public class EnemyAttackG implements PaintingInterface {

private Img \_attackImg;

private int \_width;

private int \_height;

private String[] \_pathArr = { "Pawns\\Weapons\\Animations\\Melee\_Combat1.png",

"Pawns\\Weapons\\Animations\\Melee\_Combat2.png", "Pawns\\Weapons\\Animations\\Melee\_Combat3.png",

"Pawns\\Weapons\\Animations\\Melee\_Combat4.png", "Pawns\\Weapons\\Animations\\Melee\_Combat5.png" };

private int \_turn = 0;

private double \_angle;

private MeleeAttackInterface \_listener;

/\*\*

\* Constructor

\* @param width - the width of the attack (in pixels)

\* @param height - the height of the attack (in pixels)

\* @param angle - the angle of direction of the attack

\* @param listener - the EnemyAttack that listens for the end of animation

\*/

public EnemyAttackG(int width, int height, double angle, MeleeAttackInterface listener) {

\_listener = listener;

\_width = width;

\_height = height;

\_angle = angle;

\_attackImg = new Img();

// \_attackImg.setImage(\_pathArr[0]);

\_attackImg.setImgCords(0, 0);

\_attackImg.setImgSize(\_width, \_height);

// \_attackImg.setImage(ImageTools.rotate(\_attackImg, \_angle));

}

/\*\*

\* get the next img panel or if finished notify the listener about it

\*/

public void nextAttackImg() {

if (\_turn >= \_pathArr.length) {

\_listener.meleeAttackFinished();

} else {

\_attackImg.setImage(\_pathArr[\_turn++]);

\_attackImg.setImage(ImageTools.rotate(\_attackImg, \_angle));

}

}

/\*\*

\* paint component for the attackImg

\*/

@Override

public void myPaintComponent(Graphics g) {

\_attackImg.drawImg(g);

}

}

**unit.enemy.EnemyG**

package unit.enemy;

import java.awt.Graphics;

import images.Img;

import ultilityTools.ImageTools;

import ultilityTools.PaintingInterface;

import unit.player.ApparelSet;

import unit.player.PlayerG;

/\*\*

\* EnemyG is a class that represents the graphical appearance of an enemy on a map

\* @author Peter

\*

\*/

public class EnemyG implements PaintingInterface{

private int \_unitHeight;

private int \_unitWidth;

private Img \_unit;

private ApparelSet \_unitSet;

private int \_direction;

/\*\*

\* Constructor

\* @param unitHeight - the height of the Enemy Panel

\* @param unitWidth - the width of the Enemy Panel

\*/

public EnemyG(int unitHeight, int unitWidth) {

\_unitHeight = unitHeight;

\_unitWidth = unitWidth;

\_unitSet = new ApparelSet();

\_unitSet.setFile("Pawns\\Aliens\\Terminator\_Male\_front.png",false);

\_unit = new Img();

setImgSide(0);

setAllSizes(\_unitHeight, \_unitWidth);

}

/\*\*

\* set a new Size for the enemy

\* @param unitHeight - the new Height of the enemy

\* @param unitWidth - the new Width of the enemy

\*/

public void setAllSizes(int unitHeight, int unitWidth) {

\_unitHeight = unitHeight;

\_unitWidth = unitWidth;

\_unit.setImgSize(\_unitWidth, \_unitHeight);

\_unit.setImgCords(0, 0);

}

/\*\*

\* select the proper img based on the side the enemy is facing

\*

\* @param side

\* - the side that the enemy is facing (0 front , 1 right , 2 back ,3 left)

\*/

public void setImgSide(int side) {// 0 front , 1 right , 2 back ,3 left

\_direction = side;

/\*\* if the direction is sideway \*/

if (\_direction % 2 == 1) {

\_unit.setImage(\_unitSet.getSide());

}

/\*\* if the direction is back \*/

else if (\_direction == 2) {

\_unit.setImage(\_unitSet.getBack());

}

/\*\* if the direction is front \*/

else if (\_direction == 0) {

\_unit.setImage(\_unitSet.getFront());

}

}

/\*\*

\* the implemented method of the paintingInterface , so the Player would appear on map

\*/

@Override

public void myPaintComponent(Graphics g) {

switch (\_direction) {

case 0:

case 1:

case 2:

\_unit.drawImg(g);

break;

case 3:

/\*\* if the direction is left flip the image \*/

\_unit.drawImgVertically(g);

break;

}

}

}

**unit.enemy.MeleeAttackInterface**

package unit.enemy;

/\*\*

\* Melee attack interface is used by the Enemy Attack to notify the attack when the animation is finished

\* @author Peter

\*

\*/

public interface MeleeAttackInterface {

public void meleeAttackFinished();

}

**unit.player.ApparelSet**

package unit.player;

import java.io.File;

import java.util.LinkedList;

/\*\*

\* ApparelSet is a class that represents a certain ware(clothing) at all directions

\*

\* @author Peter

\*/

public class ApparelSet {

private LinkedList<String> \_allFrontSets;

private int \_currentSetIndex;

private String \_front;

private String \_back;

private String \_side;

/\*\*

\* constructor empty sets

\*/

public ApparelSet() {

this(null, null, null);

}

/\*\*

\* Constructor

\*

\* @param front

\* - front Img Path

\* @param back

\* - back Img Path

\* @param side

\* - side Img Path

\*/

public ApparelSet(String front, String back, String side) {

\_currentSetIndex = 0;

\_front = front;

\_back = back;

\_side = side;

}

/\*\*

\* adds to allFrontSets all of the items on the list with the previous path that is received

\*

\* @param prePath

\* - the previous path that is received

\* @param frontPaths

\* - list that contains Strings that path to a front img

\*/

public void addFrontSet(String prePath, LinkedList<String> frontPaths) {

\_allFrontSets = new LinkedList<String>();

\_currentSetIndex = 0;

for (int i = 0; i < frontPaths.size(); i++) {

\_allFrontSets.add(prePath + frontPaths.get(i));

}

}

/\*\*

\* which front img is being used right now by comparing it to every single one and setting the

\* currentSetIndex to it.

\*/

public void findCurrentSetIndex() {

for (int i = 0; i < \_allFrontSets.size(); i++) {

if (\_allFrontSets.get(i).equals(\_front)) {

\_currentSetIndex = i;

return;

}

}

}

/\*\*

\* set to default (first) option for both weapons or clothing.

\*/

public void resetCurrent() {

\_front = new String(\_allFrontSets.getFirst());

if (!\_front.contains("Weapons")) {

setFront(\_allFrontSets.get(\_currentSetIndex), true);

putMatching(0);

} else

setMatchingNoDirections(\_allFrontSets.get(\_currentSetIndex));

System.out.println(\_front);

}

/\*\*

\* exclusivly used for weapons since i have only 1 model for a weapon

\*

\* @param file

\* - the file name

\*/

public void setMatchingNoDirections(String file) {

File f = new File("srcImages\\" + file);

if (f.isFile()) {

\_front = new String(file);

\_back = new String(file);

\_side = new String(file);

}

}

/\*\*

\* picking next item on the list or previous depends on the variable received

\*

\* @param isNext

\* - if true than pick the next item on list , else pick the previous.

\*/

public void switchTo(boolean isNext) {

if (isNext) {

\_currentSetIndex = (\_currentSetIndex + 1) % \_allFrontSets.size();

} else {

if (\_currentSetIndex == 0)

\_currentSetIndex = \_allFrontSets.size();

\_currentSetIndex = (\_currentSetIndex - 1) % \_allFrontSets.size();

}

System.out.println("Apperal " + \_allFrontSets.get(\_currentSetIndex));

if (!\_front.contains("Weapons")) {

setFront(\_allFrontSets.get(\_currentSetIndex), true);

putMatching(0);

} else

setMatchingNoDirections(\_allFrontSets.get(\_currentSetIndex));

}

/\*\*

\* the function puts matching ImgPaths for the side that is forced.0-front,1-back,2-side

\*

\* @param force

\* - the force decides who exactly to force the check

\*/

public void putMatching(int force) {

if (\_front != null && force == 0) {

\_back = new String(\_front.substring(0, \_front.lastIndexOf('\_')) + "\_back.png");

\_side = new String(\_front.substring(0, \_front.lastIndexOf('\_')) + "\_side.png");

return;

}

if (\_back != null && force == 1) {

\_front = new String(\_back.substring(0, \_back.lastIndexOf('\_')) + "\_front.png");

\_side = new String(\_back.substring(0, \_back.lastIndexOf('\_')) + "\_side.png");

return;

}

if (\_side != null && force == 2) {

\_back = new String(\_side.substring(0, \_side.lastIndexOf('\_')) + "\_back.png");

\_front = new String(\_side.substring(0, \_side.lastIndexOf('\_')) + "\_front.png");

return;

}

}

/\*\*

\* the function checks if all 3 pieces exist , and if they String Paths contain the proper

\* endings

\*

\* @return true if everyting is fine ,else false

\*/

public boolean checkIfSamePiece() {

if (\_front == null || \_back == null || \_side == null)

return false;

File frontFile = new File(\_front);

File backFile = new File(\_back);

File sideFile = new File(\_side);

if (!frontFile.exists() || !backFile.exists() || !sideFile.exists())

return false;

String frontName = frontFile.getName();

String backName = backFile.getName();

String sideName = sideFile.getName();

String subStrArr[] = { frontName.substring(0, frontName.lastIndexOf('\_')),

backName.substring(0, backName.lastIndexOf('\_')), sideName.substring(0, sideName.lastIndexOf('\_')) };

return (subStrArr[0] == subStrArr[1] && subStrArr[1] == subStrArr[2]);

}

/\*\*

\* the function receives a file , doesnt know which ending it has. decides which set to do ,

\* based on the ending of the file path

\*

\* @param file

\* - the file path of the img

\*/

public void setFile(String file, boolean isOutfit) {

if (file.endsWith("\_front.png"))

setFront(file, isOutfit);

else if (file.endsWith("\_back.png"))

setBack(file);

else if (file.endsWith("\_side.png"))

setSide(file);

else

System.out.println(file + " BAD FILE");

}

/\*\*

\* set the front Image,forces matching for front

\*

\* @param front

\* - the front imgPath

\*/

public void setFront(String front, boolean isOutfit) {

\_front = front;

if (isOutfit)

findCurrentSetIndex();

putMatching(0);

}

/\*\*

\* set the back Image,forces matching for back

\*

\* @param back

\* - the back imgPath

\*/

public void setBack(String back) {

\_back = back;

putMatching(1);

}

/\*\*

\* set the side Image,forces matching for side

\*

\* @param side

\* - the side imgPath

\*/

public void setSide(String side) {

\_side = side;

putMatching(2);

}

/\*\*

\* @return the front direction img path

\*/

public String getFront() {

return \_front;

}

/\*\*

\* @return the back direction img path

\*/

public String getBack() {

return \_back;

}

/\*\*

\* @return the side direction img path

\*/

public String getSide() {

return \_side;

}

/\*\*

\* @return the list containing all front direction img paths

\*/

public LinkedList<String> getAllFrontSets() {

return \_allFrontSets;

}

}

**unit.player.Outfit**

package unit.player;

import ultilityTools.FileChooser;

import ultilityTools.FileFinder;

import unit.customizeScreen.CustomizeScreen;

/\*\*

\* Outfit represents the general outfit of a character

\*

\* @author Peter

\*/

public class Outfit {

private FileChooser \_fc;

private ApparelSet \_figure;

private ApparelSet \_face;

private ApparelSet \_bottom;

private ApparelSet \_top;

private ApparelSet \_hair;

private ApparelSet \_headgear;

private ApparelSet \_weapon;

/\*\*

\* Constructor prepares file selector ( will be replaced with selection panel) initialitezes all

\* apparel sets

\*/

public Outfit() {

\_fc = new FileChooser();

\_figure = new ApparelSet();

\_face = new ApparelSet();

\_bottom = new ApparelSet();

\_top = new ApparelSet();

\_hair = new ApparelSet();

\_headgear = new ApparelSet();

\_weapon = new ApparelSet();

}

/\*\*

\* fills all of the lists with a selection from the Clothing\Weapons folder that all ends with front

\*/

public void fillAllApparels() {

\_figure.addFrontSet("Pawns\\Humans\\", FileFinder.getAllFiles("srcImages\\Pawns\\Humans\\", "\_front.png"));

\_face.addFrontSet("Pawns\\Clothing\\Faces\\",

FileFinder.getAllFiles("srcImages\\Pawns\\Clothing\\Faces", "\_front.png"));

\_bottom.addFrontSet("Pawns\\Clothing\\Bottom\\",

FileFinder.getAllFiles("srcImages\\Pawns\\Clothing\\Bottom", "\_front.png"));

\_top.addFrontSet("Pawns\\Clothing\\Top\\",

FileFinder.getAllFiles("srcImages\\Pawns\\Clothing\\Top", "\_front.png"));

\_hair.addFrontSet("Pawns\\Clothing\\Hair\\",

FileFinder.getAllFiles("srcImages\\Pawns\\Clothing\\Hair", "\_front.png"));

\_headgear.addFrontSet("Pawns\\Clothing\\HeadGear\\",

FileFinder.getAllFiles("srcImages\\Pawns\\Clothing\\HeadGear", "\_front.png"));

if (CustomizeScreen.getInstance() == null)

\_weapon.addFrontSet("Pawns\\Weapons\\Rifles\\",

FileFinder.getAllFiles("srcImages\\Pawns\\Weapons\\Rifles", ""));

else

\_weapon.addFrontSet("Pawns\\Weapons\\" + CustomizeScreen.getInstance().getWeaponSelected() + "\\",

FileFinder.getAllFiles(

"srcImages\\Pawns\\Weapons\\" + CustomizeScreen.getInstance().getWeaponSelected(), ""));

System.out.println(\_weapon.getAllFrontSets());

}

/\*\*

\* Selects the files for each apparel accordiongly

\*/

/\*

\* public void SelectAllApparels() {

\* setFigureFile("Pawns\\Humans\\" + \_fc.choosePngFile("srcImages\\Pawns\\Humans").getName());

\* setFaceFile("Pawns\\Clothing\\Faces\\" + \_fc.choosePngFile("srcImages\\Pawns\\Clothing\\Faces

\* ").getName());

\* setBottomFile("Pawns\\Clothing\\Bottom\\" + \_fc.choosePngFile("srcImages\\Pawns\\Clothing\\

\* Bottom").getName());

\* setTopFile("Pawns\\Clothing\\Top\\" + \_fc.choosePngFile("srcImages\\Pawns\\Clothing\\Top").

\* getName());

\* setHairFile("Pawns\\Clothing\\Hair\\" + \_fc.choosePngFile("srcImages\\Pawns\\Clothing\\Hair")

\* .getName()); setHeadgearFile(

\* "Pawns\\Clothing\\HeadGear\\" + \_fc.choosePngFile("srcImages\\Pawns\\Clothing\\HeadGear").

\* getName()); // setWeaponFile(weaponFile); //

\* \_weapon.setSide(\_fc.choosePngFile("src\\images\\srcImages\\Pawns\\Weapons").getPath()); }

\*/

/\*\*

\* Set default apearal

\*/

public void SelectAllDefault() {

setFigureFile("Pawns\\Humans\\Hum1\_front.png");

setFaceFile("Pawns\\Clothing\\Faces\\Face1\_front.png");

setBottomFile("Pawns\\Clothing\\Bottom\\MilitaryPants\_Male\_front.png");

setTopFile("Pawns\\Clothing\\Top\\Combat\_Male\_front.png");

setHairFile("Pawns\\Clothing\\Hair\\crisis\_front.png");

setHeadgearFile("Pawns\\Clothing\\HeadGear\\Addyhat\_front.png");

setWeaponFile("Pawns\\Weapons\\Rifles\\MD50.png");

}

/\*\*

\* set a new Figure instead and isOutfit is true since its not weapon related.

\* @param figureFile - the new figure file path

\*/

public void setFigureFile(String figureFile) {

\_figure.setFile(figureFile, true);

}

/\*\*

\* set a new Face instead and isOutfit is true since its not weapon related.

\* @param faceFile - the new face file path

\*/

public void setFaceFile(String faceFile) {

\_face.setFile(faceFile, true);

}

/\*\*

\* set a new Bottom instead and isOutfit is true since its not weapon related.

\* @param bottomFile -the new bottom file path

\*/

public void setBottomFile(String bottomFile) {

\_bottom.setFile(bottomFile, true);

}

/\*\*

\* set a new top instead and isOutfit is true since its not weapon related.

\* @param topFile -the new top file path

\*/

public void setTopFile(String topFile) {

\_top.setFile(topFile, true);

}

/\*\*

\* set a new hair instead and isOutfit is true since its not weapon related.

\* @param hairFile -the new hair file path

\*/

public void setHairFile(String hairFile) {

\_hair.setFile(hairFile, true);

}

/\*\*

\* set a new headgear instead and isOutfit is true since its not weapon related.

\* @param headgearFile -the new headgear file path

\*/

public void setHeadgearFile(String headgearFile) {

\_headgear.setFile(headgearFile, true);

}

/\*\*

\* set a new weapon instead and isOutfit

\* @param weaponFile -the new weapon file path

\*/

public void setWeaponFile(String weaponFile) {

\_weapon.setMatchingNoDirections(weaponFile);

}

/\*\*

\* @return the FileChooser that helps select files

\*/

public FileChooser getFc() {

return \_fc;

}

/\*\*

\* @return the figure set

\*/

public ApparelSet getFigure() {

return \_figure;

}

/\*\*

\* @return the face set

\*/

public ApparelSet getFace() {

return \_face;

}

/\*\*

\* @return the bottom set

\*/

public ApparelSet getBottom() {

return \_bottom;

}

/\*\*

\* @return the top set

\*/

public ApparelSet getTop() {

return \_top;

}

/\*\*

\* @return the hair set

\*/

public ApparelSet getHair() {

return \_hair;

}

/\*\*

\* @return the headgear set

\*/

public ApparelSet getHeadgear() {

return \_headgear;

}

/\*\*

\* @return the weapon set

\*/

public ApparelSet getWeapon() {

return \_weapon;

}

}

**unit.player.Player**

package unit.player;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.FocusListener;

import java.awt.event.KeyEvent;

import java.util.LinkedList;

import javax.swing.JDialog;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.Timer;

import engine.Unit;

import engine.input.KeyboardInput;

import mainInitialize.GameStats;

import map.MapG;

import map.MapGFrame;

import map.MapGPanel;

import unit.FireCooldown;

import unit.MovingUnit;

import unit.WinningInterface;

import unit.ObjectOnMap;

import unit.Projectile;

import unit.customizeScreen.DisplayScreen;

/\*\*

\* Player represents a player which extends ObjectOnMap because it is on map

\*

\* @author Peter

\*/

public class Player extends MovingUnit implements ActionListener, WinningInterface {

/\*\* the stats of the player \*/

private Unit \_stats;

/\*\* the graphic class of the player \*/

private PlayerG \_graphics;

/\*\*

\* if the player is busy doing one of the actions the flag will be active

\*/

private boolean \_rightBusy = false;

private boolean \_leftBusy = false;

private boolean \_upBusy = false;

private boolean \_downBusy = false;

private boolean \_busy = false;

private boolean \_isBlocked;

/\*\*

\* Constructor

\*

\* @param x

\* - the graphical X position of the player

\* @param y

\* - the graphical Y position of the player

\* @param width

\* - the width of the player

\* @param height

\* - the height of the player

\* @param map

\* - the Graphical map

\*/

public Player(int x, int y, int width, int height) {

/\*\* use Object's on map constructor \*/

super(x - PlayerG.padding, y - PlayerG.padding, width + PlayerG.padding \* 2, height + PlayerG.padding \* 2);

/\*\* initialize stats \*/

\_stats = new Unit('P', x / 64, y / 64, 100, this);

/\*\* add player as listener \*/

KeyboardInput.getInstance().addListener(this);

/\*\* set focus on player \*/

// MapGFrame.getInstance().setFocusable(true);

// MapGFrame.getInstance().setFocusTraversalKeysEnabled(false);

/\*\* create new Graphics \*/

setCustomGraphics(height, width);

/\*\* the action timer \*/

\_timer = new Timer(20, this);

\_timer.start();

\_fireListener = new FireCooldown();

\_fireTimer = new Timer(20, \_fireListener);

\_fireTimer.start();

}

/\*\*

\* Sets the graphics for the player based on the DisplayScreen ( if it exists) while also

\* matching the size and setting this panel as parent

\*

\* @param height-

\* the height of the Player (in pixels) and sets the size to it

\* @param width

\* - the width of the Player (in pixels) and sets the size to it

\*/

public void setCustomGraphics(int height, int width) {

if (DisplayScreen.getInstance() == null || DisplayScreen.getInstance().getPlayer() == null)

\_graphics = new PlayerG(height, width);

else {

\_graphics = DisplayScreen.getInstance().getPlayer();

}

\_graphics.setParent(this);

\_graphics.setAllSizes(height, width);

setObjectGraphicClass(\_graphics);

}

/\*\*

\* convert Boolean value to integer value

\*

\* @param bool

\* - the boolean value

\* @return 1 if bool is true else 0

\*/

public int convertBooleanToInt(boolean bool) {

return (bool) ? 1 : 0;

}

/\*\*

\* The action that is done after every tick checks if the player won , checks if he is busy if

\* not checks if he shoots Else check if he moves

\*/

@Override

public void actionPerformed(ActionEvent e) {

LinkedList<Boolean> keys = KeyboardInput.getInstance().get\_keys();

actionIfWin();

if (!\_busy) {

if (keys.get(KeyEvent.VK\_RIGHT))

attackRight();

else if (keys.get(KeyEvent.VK\_LEFT))

attackLeft();

else if (keys.get(KeyEvent.VK\_UP))

attackUp();

else if (keys.get(KeyEvent.VK\_DOWN))

attackDown();

}

if (\_fireListener.getFireCooldown() == 0) {

/\*\*

\* if 'A' is held and none of the actions are done or the left action is currently busy

\*/

if ((keys.get(KeyEvent.VK\_A) && !\_busy) || (\_leftBusy && \_busy)) {

\_leftBusy = true;

\_busy = true;

if (\_i == 0) {

\_isBlocked = isBlocked(\_stats.getCurrentX() - 1, \_stats.getCurrentY());

/\*\* if the location player moves is floor \*/

if (!\_isBlocked)

\_stats.setCurrentX(\_stats.getCurrentX() - 1);

GameStats.addOccupied(\_stats.getCurrentX(), \_stats.getCurrentY());

}

/\*\* Move left ,receives boolean that tells which animation cycle to do \*/

moveLeft(\_isBlocked);

/\*\* set direction in order to draw Properly \*/

\_graphics.setImgSide(3);

/\*\* after finishing cycle \*/

if (\_i == 16) {

if (!\_isBlocked)

GameStats.removeOccupied(\_stats.getCurrentX() + 1, \_stats.getCurrentY());

/\*\* reset the \_i and free the busy flags \*/

\_i = 0;

\_leftBusy = false;

\_busy = false;

\_isBlocked = false;

}

// repaint();

return;

}

/\*\*

\* if 'D' is held and none of the actions are done or the right action is currently busy

\*/

if (((keys.get(KeyEvent.VK\_D) && !\_busy)) || (\_rightBusy && \_busy)) {

\_rightBusy = true;

\_busy = true;

if (\_i == 0) {

\_isBlocked = isBlocked(\_stats.getCurrentX() + 1, \_stats.getCurrentY());

/\*\* if the location player moves is floor \*/

if (!\_isBlocked)

\_stats.setCurrentX(\_stats.getCurrentX() + 1);

GameStats.addOccupied(\_stats.getCurrentX(), \_stats.getCurrentY());

}

/\*\* Move right ,receives boolean that tells which animation cycle to do \*/

moveRight(\_isBlocked);

/\*\* set direction in order to draw Properly \*/

\_graphics.setImgSide(1);

/\*\* after finishing cycle \*/

if (\_i == 16) {

if (!\_isBlocked)

GameStats.removeOccupied(\_stats.getCurrentX() - 1, \_stats.getCurrentY());

/\*\* reset the \_i and free the busy flags \*/

\_i = 0;

\_rightBusy = false;

\_busy = false;

\_isBlocked = false;

}

// repaint();

return;

}

/\*\*

\* if 'W' is held and none of the actions are done or the up action is currently busy

\*/

if (((keys.get(KeyEvent.VK\_W) && !\_busy)) || (\_upBusy && \_busy)) {

\_upBusy = true;

\_busy = true;

if (\_i == 0) {

\_isBlocked = isBlocked(\_stats.getCurrentX(), \_stats.getCurrentY() - 1);

/\*\* if the location player moves is floor \*/

if (!\_isBlocked)

\_stats.setCurrentY(\_stats.getCurrentY() - 1);

GameStats.addOccupied(\_stats.getCurrentX(), \_stats.getCurrentY());

}

/\*\* Move up , receives boolean that tells which animation cycle to do \*/

moveUp(\_isBlocked);

/\*\* set direction in order to draw Properly \*/

\_graphics.setImgSide(2);

/\*\* after finishing cycle \*/

if (\_i == 16) {

if (!\_isBlocked)

GameStats.removeOccupied(\_stats.getCurrentX(), \_stats.getCurrentY() + 1);

/\*\* reset the \_i and free the busy flags \*/

\_i = 0;

\_upBusy = false;

\_busy = false;

\_isBlocked = false;

}

// repaint();

return;

}

/\*\*

\* if 'S' is held and none of the actions are done or the down action is currently busy

\*/

if (((keys.get(KeyEvent.VK\_S) && !\_busy)) || (\_downBusy && \_busy)) {

\_downBusy = true;

\_busy = true;

if (\_i == 0) {

\_isBlocked = isBlocked(\_stats.getCurrentX(), \_stats.getCurrentY() + 1);

/\*\* if the location player moves is floor \*/

if (!\_isBlocked)

\_stats.setCurrentY(\_stats.getCurrentY() + 1);

GameStats.addOccupied(\_stats.getCurrentX(), \_stats.getCurrentY());

}

/\*\* Move down , receives boolean that tells which animation cycle to do \*/

moveDown(\_isBlocked);

/\*\* set direction in order to draw Properly \*/

\_graphics.setImgSide(0);

/\*\* after finishing cycle \*/

if (\_i == 16) {

if (!\_isBlocked)

GameStats.removeOccupied(\_stats.getCurrentX(), \_stats.getCurrentY() - 1);

/\*\* reset the \_i and free the busy flags \*/

\_i = 0;

\_downBusy = false;

\_busy = false;

\_isBlocked = false;

}

// repaint();

return;

}

}

if (\_graphics.getDirection() != 0) {

\_graphics.setImgSide(0);

repaint();

}

}

/\*\*

\* check if the player won , and notify the user about it

\*/

public void actionIfWin() {

if (isWin()) {

JDialog jd;

jd = new JDialog(MapGFrame.getInstance(), "Test Dialog");

JPanel winPanel = new JPanel();

winPanel.add(new JLabel("Player Won!"));

jd.add(winPanel);

jd.pack();

jd.setVisible(true);

GameStats.pauseGame();

}

}

/\*\*

\* check if the player is the only one left in the global unit list

\*/

public boolean isWin() {

return (GameStats.getPlayers().size() == 1 && GameStats.getPlayers().getFirst() instanceof Player);

}

/\*\*

\* @return player graphics of the player

\*/

public PlayerG getPlayerGraphics() {

return \_graphics;

}

/\*\*

\* @return Unit Stats of the player

\*/

public Unit getStats() {

return \_stats;

}

/\*\*

\* pauses the unit , used for pausing the game

\*/

@Override

public void pauseUnit() {

\_timer.stop();

\_fireTimer.stop();

}

/\*\*

\* starts the unit, used for continueing the game

\*/

@Override

public void startUnit() {

\_timer.start();

\_fireTimer.start();

}

/\*\*

\* launch a projectile to the right direction if the fireCooldown is 0

\*/

@Override

public void attackRight() {

if (\_fireListener.getFireCooldown() == 0) {

System.out.println("Projectile started");

\_fireListener.setFireCooldown(20);// 10 \*20;

Projectile p = new Projectile(getRealX() + 16, getRealY(), 64, 64, 16, 180, 90.0);

}

}

/\*\*

\* launch a projectile to the left direction if the fireCooldown is 0

\*/

@Override

public void attackLeft() {

if (\_fireListener.getFireCooldown() == 0) {

System.out.println("Projectile started");

\_fireListener.setFireCooldown(20);// 10 \*20;

Projectile p = new Projectile(getRealX() - 16, getRealY(), 64, 64, 16, 0, 270.0);

}

}

/\*\*

\* launch a projectile to the upper direction if the fireCooldown is 0

\*/

@Override

public void attackUp() {

if (\_fireListener.getFireCooldown() == 0) {

System.out.println("Projectile started");

\_fireListener.setFireCooldown(20);// 10 \*20;

Projectile p = new Projectile(getRealX(), getRealY() - 16, 64, 64, 16, 90, 180.0);

}

}

/\*\*

\* launch a projectile to the below direction if the fireCooldown is 0

\*/

@Override

public void attackDown() {

if (\_fireListener.getFireCooldown() == 0) {

System.out.println("Projectile started");

\_fireListener.setFireCooldown(20);// 10 \*20;

Projectile p = new Projectile(getRealX(), getRealY() + 16, 64, 64, 16, 270, 0.0);

}

}

/\*\*

\* @returns the ObjectX without the padding ( in pixels)

\*/

@Override

public int getRealX() {

return getObjectX() + PlayerG.padding;

}

/\*\*

\* @returns the ObjectY without the padding ( in pixels)

\*/

@Override

public int getRealY() {

return getObjectY() + PlayerG.padding;

}

/\*\*

\* @returns the Object Height without the padding ( in pixels)

\*/

@Override

public int getRealHeight() {

return getObjectHeight() - PlayerG.padding \* 2;

}

/\*\*

\* @returns the Object Width without the padding ( in pixels)

\*/

@Override

public int getRealWidth() {

return getObjectWidth() - PlayerG.padding \* 2;

}

}

**unit.player.PlayerG**

package unit.player;

import java.awt.Graphics;

import javax.swing.JPanel;

import images.Img;

import ultilityTools.ImageTools;

import ultilityTools.PaintingInterface;

import unit.customizeScreen.CustomizeInterface;

/\*\*

\* UnitG represents the graphical class of the unit or player

\*

\* @author Peter

\*/

public class PlayerG implements PaintingInterface, CustomizeInterface {

private int \_unitHeight;

private int \_unitWidth;

private Outfit \_customize;

private Img \_unit;

private Img \_bottom;

private Img \_top;

private Img \_headGear;

private Img \_hair;

private Img \_weapon;

private Img \_face;

private int \_direction;

/\*\*

\* the parent is needed inorder to switch between customize screen and game

\*/

private JPanel \_parent;

/\*\*

\* padding is essential for the player if we dont want some of the models to be cut.

\*/

public final static int padding = 10;

/\*\*

\* Constructor

\*

\* @param unitHeight

\* - the height of the player

\* @param unitWidth

\* - the width of the player

\*/

public PlayerG(int unitHeight, int unitWidth) {

\_unitHeight = unitHeight;

\_unitWidth = unitWidth;// \\images\\srcImages\\Pawns\\Clothing\\Bottom\\Walls\\0001Wall\_Atlas.png

/\*\* initialize outfit set \*/

\_customize = new Outfit();

/\*\* allow user to select the outfits \*/

// \_customize.SelectAllApparels();

\_customize.fillAllApparels();

\_customize.SelectAllDefault();

/\*\* initialize img with blank images \*/

\_unit = new Img();

\_face = new Img();

\_bottom = new Img();

\_top = new Img();

\_headGear = new Img();

\_weapon = new Img();

\_hair = new Img();

setAllSizes(\_unitHeight, \_unitWidth);

setImgSide(0);

}

/\*\*

\* Constructor

\*

\* @param unitHeight

\* - the height of the player

\* @param unitWidth

\* - the width of the player

\* @param parent

\* - the Panel parent of this graphics class

\*/

public PlayerG(int unitHeight, int unitWidth, JPanel parent) {

this(unitHeight, unitWidth);

\_parent = parent;

}

/\*\*

\* Sets all size \ positions based on the received height and width

\*

\* @param unitHeight

\* - the new height of the Player

\* @param unitWidth

\* - the new width of the player

\*/

public void setAllSizes(int unitHeight, int unitWidth) {

\_unitHeight = unitHeight;

\_unitWidth = unitWidth;

\_unit.setImgSize(\_unitWidth, \_unitHeight);

\_unit.setImgCords(padding, padding);

\_face.setImgSize(\_unitWidth, \_unitHeight);

\_face.setImgCords(padding, padding + (int) ((-15 / 64.0) \* \_unitHeight));

\_bottom.setImgSize(\_unitWidth, \_unitHeight);

\_bottom.setImgCords(padding, padding);

\_top.setImgSize(\_unitWidth, \_unitHeight);

\_top.setImgCords(padding, padding + (int) ((5 / 64.0) \* \_unitHeight));

\_headGear.setImgSize(\_unitWidth, \_unitHeight);

\_headGear.setImgCords(padding, padding + (int) ((-18 / 64.0) \* \_unitHeight));

\_weapon.setImgSize(\_unitWidth, \_unitHeight);

\_weapon.setImgCords(padding + (int) ((10 / 64.0) \* \_unitWidth), padding + (int) ((8 / 64.0) \* \_unitHeight));

\_hair.setImgSize(\_unitWidth, \_unitHeight);

\_hair.setImgCords(padding + (int) ((-5 / 64.0)), padding + (int) ((-15 / 64.0) \* \_unitHeight));

if (\_parent != null) {

\_parent.repaint();

}

}

/\*\*

\* select the proper img based on the side the player is facing

\*

\* @param side

\* - the side that the player is facing (0 front , 1 right , 2 back ,3 left)

\*/

public void setImgSide(int side) {// 0 front , 1 right , 2 back ,3 left

\_direction = side;

/\*\* if the direction is sideway \*/

if (\_direction % 2 == 1) {

\_unit.setImage(\_customize.getFigure().getSide());

\_face.setImage(\_customize.getFace().getSide());

\_bottom.setImage(\_customize.getBottom().getSide());

\_top.setImage(\_customize.getTop().getSide());

\_headGear.setImage(\_customize.getHeadgear().getSide());

// \_weapon.setImage(\_customize.getWeapon().getSide());

\_hair.setImage(\_customize.getHair().getSide());

}

/\*\* if the direction is back \*/

else if (\_direction == 2) {

\_unit.setImage(\_customize.getFigure().getBack());

\_face.setImage(\_customize.getFace().getBack());

\_bottom.setImage(\_customize.getBottom().getBack());

\_top.setImage(\_customize.getTop().getBack());

\_headGear.setImage(\_customize.getHeadgear().getBack());

// \_weapon.setImage(\_customize.getWeapon().getBack());

\_hair.setImage(\_customize.getHair().getBack());

}

/\*\* if the direction is front \*/

else if (\_direction == 0) {

\_unit.setImage(\_customize.getFigure().getFront());

\_face.setImage(\_customize.getFace().getFront());

\_bottom.setImage(\_customize.getBottom().getFront());

\_top.setImage(\_customize.getTop().getFront());

\_headGear.setImage(\_customize.getHeadgear().getFront());

\_hair.setImage(\_customize.getHair().getFront());

}

\_weapon.setImage(\_customize.getWeapon().getFront());

}

/\*\*

\* Set a new Parent for the graphics class

\*

\* @param parent

\* - the new Panel Parent

\*/

public void setParent(JPanel parent) {

\_parent = parent;

}

/\*\*

\* the implemented method of the paintingInterface , so the Player would appear on map

\*/

@Override

public void myPaintComponent(Graphics g) {

switch (\_direction) {

case 0:

\_unit.drawImg(g);

\_bottom.drawImg(g);

\_top.drawImg(g);

\_weapon.setImgCords(padding + (int) ((5 / 64.0) \* \_unitWidth), padding + (int) ((8 / 64.0) \* \_unitHeight));

\_weapon.drawImg(g);

\_unit.drawPartImage(g, 0, 0, (int) ((\_unitHeight) / 2.5), \_unitWidth);

\_face.drawImg(g);

\_hair.drawImg(g);

\_headGear.drawImg(g);

break;

case 1:

\_weapon.setImgCords(padding + (int) ((12 / 64.0) \* \_unitWidth), padding + (int) ((8 / 64.0) \* \_unitHeight));

\_weapon.drawImgRotate(g, 85);

\_unit.drawImg(g);

// \_face.drawImg(g);

\_bottom.drawImg(g);

\_top.drawImg(g);

// \_hair.drawImg(g);

// \_headGear.drawImg(g);

\_unit.drawPartImage(g, 0, 0, (int) ((\_unitHeight) / 2.5), \_unitWidth);

\_face.drawImg(g);

\_hair.drawImg(g);

\_headGear.drawImg(g);

break;

case 2:

\_weapon.setImgCords(padding + (int) ((5 / 64.0) \* \_unitWidth), padding + (int) ((-5 / 64.0) \* \_unitHeight));

\_weapon.drawImg(g);

\_unit.drawImg(g);

\_bottom.drawImg(g);

\_top.drawImg(g);

\_unit.drawPartImage(g, 0, 0, (int) ((\_unitHeight) / 3), \_unitWidth);

\_face.drawImg(g);

\_hair.drawImg(g);

\_headGear.drawImg(g);

break;

case 3:

/\*\* if the direction is left flip the image \*/

\_weapon.setImgCords(padding + (int) ((-12 / 64.0) \* \_unitWidth),

padding + (int) ((8 / 64.0) \* \_unitHeight));

\_weapon.setImage(ImageTools.rotate(\_weapon, 85));

\_weapon.drawImgVertically(g);

\_unit.drawImgVertically(g);

\_bottom.drawImgVertically(g);

\_top.drawImgVertically(g);

\_unit.drawPartImageVeritcally(g, 0, 0, (int) ((\_unitHeight) / 2.5), \_unitWidth);

\_face.drawImgVertically(g);

\_hair.drawImgVertically(g);

\_headGear.drawImgVertically(g);

break;

}

}

/\*\*

\* whenever a button is clicked in the customize menu

\*/

@Override

public void CustomizeButtonClicked(String name, boolean isNext) {

if (name.compareTo("Headgear") == 0) {

\_customize.getHeadgear().switchTo(isNext);

} else if (name.compareTo("Hair") == 0) {

\_customize.getHair().switchTo(isNext);

} else if (name.compareTo("Face") == 0) {

\_customize.getFace().switchTo(isNext);

} else if (name.compareTo("Body") == 0) {

\_customize.getFigure().switchTo(isNext);

} else if (name.compareTo("Top") == 0) {

\_customize.getTop().switchTo(isNext);

} else if (name.compareTo("Weapon") == 0) {

\_customize.getWeapon().switchTo(isNext);

} else if (name.compareTo("Bottom") == 0) {

\_customize.getBottom().switchTo(isNext);

}

setImgSide(0);

\_parent.repaint();

}

/\*\*

\* Copies a new players graphics to the current players graphics

\*

\* @param playerGraphics

\* - the new Players graphics

\*/

public void copyPlayerG(PlayerG playerGraphics) {

\_customize = playerGraphics.\_customize;

setAllSizes(playerGraphics.\_unitHeight, playerGraphics.\_unitWidth);

setImgSide(0);

}

/\*\*

\* @return - the outfit which contains all of the clothing with a weapon

\*/

public Outfit getCustomize() {

return \_customize;

}

/\*\*

\* @return - get the current direction at which the player is facing

\*/

public int getDirection() {

return \_direction;

}

}

**mainInitialize.InstructionPanel**

package mainInitialize;

import java.awt.Color;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.BoxLayout;

import javax.swing.JButton;

import javax.swing.JLabel;

import javax.swing.JPanel;

import ultilityTools.ImageTools;

/\*\*

\* this class represents the panel of instructions extends JPanel because it is panel

\*

\* @author Peter

\* @category Singleton - only contains one instance

\*/

public class InstructionPanel extends JPanel {

private JButton \_backBtn;

private JLabel \_wasdInstruction;

private JLabel \_arrowInstruction;

private JLabel \_escapeInstruction;

private static InstructionPanel singleton;

/\*\*

\* Constructor initiates buttuns and sets listener

\*/

private InstructionPanel() {

\_backBtn = ImageTools.makeButton("srcImages\\Menu\\mainMenuIcon.png");

\_wasdInstruction = ImageTools.makeLabel("srcImages\\Menu\\WasdKeyInstruction.png");

\_arrowInstruction = ImageTools.makeLabel("srcImages\\Menu\\ArrowKeyInstruction.png");

\_escapeInstruction = ImageTools.makeLabel("srcImages\\Menu\\EscapeInstruction.png");

setLayout(new BoxLayout(this, BoxLayout.Y\_AXIS));

setBackground(Color.black);

\_backBtn.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

setVisible(false);

BackgroundImgPanel.getButtonPanel().setVisible(true);

}

});

add(\_wasdInstruction);

add(\_arrowInstruction);

add(\_escapeInstruction);

add(\_backBtn);

}

/\*\*

\* initialize the singleton instance

\*

\* @return InstructionPanel only instance

\*/

public static InstructionPanel init() {

if (singleton == null)

singleton = new InstructionPanel();

return singleton;

}

/\*\*

\* get the Singleton instance

\*

\* @return InstructionPanel only instance

\*/

public static InstructionPanel getInstance() {

return singleton;

}

}

**map.BrieferScreen**

package map;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.Icon;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JDialog;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import mainInitialize.BackgroundImgPanel;

import mainInitialize.GameStats;

import mainInitialize.MainMenu;

import ultilityTools.ImageTools;

/\*\*

\* this class represents the panel of the Briefer extends JPanel because it is panel

\*

\* @author Peter

\* @category Singleton - only contains one instance

\*/

public class BrieferScreen extends JPanel {

private JButton \_restartBtn;

private JButton \_newGameBtn;

private JButton \_backToMenu;

private JLabel \_briefText;

private static BrieferScreen singleton;

private static JDialog \_parent;

/\*\*

\* Constructor sets up buttons of the victory\lose screen

\*

\* @param playerWon

\* - if true , Player won text will appear with new game option. else enemy won

\*/

private BrieferScreen(boolean playerWon) {

setBackground(Color.BLACK);

setUpLabel(playerWon);

if (playerWon) {

setUpNewGame();

}

setUpRestart();

setUpBack();

}

/\*\*

\* initialize the singleton instance

\*

\* @return BrieferScreen only instance

\*/

public static BrieferScreen init(boolean playerWon, JDialog parent) {

singleton = new BrieferScreen(playerWon);

\_parent = parent;

return singleton;

}

/\*\*

\* get the Singleton instance

\*

\* @return BrieferScreen only instance

\*/

public static BrieferScreen getInstance() {

return singleton;

}

/\*\*

\* Creates a label with the proper img

\*

\* @param playerWon-if

\* true , Player won text will appear . else enemy won text

\*/

private void setUpLabel(boolean playerWon) {

if (playerWon)

\_briefText = ImageTools.makeLabel("srcImages\\Menu\\playerWon.png");

else

\_briefText = ImageTools.makeLabel("srcImages\\Menu\\enemyWon.png");

add(\_briefText);

}

/\*\*

\* Creates Restart button that starts the game with previous map

\*/

private void setUpRestart() {

\_restartBtn = ImageTools.makeButton("srcImages\\Menu\\restartIcon.png");

\_restartBtn.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

GameStats.restartGame(false);

GameStats.startGame();

BrieferScreen.getInstance().setVisible(false);

\_parent.setVisible(false);

}

});

add(\_restartBtn);

}

/\*\*

\* Creates NewGame button that starts the game with new map

\*/

private void setUpNewGame() {

\_newGameBtn = ImageTools.makeButton("srcImages\\Menu\\startIcon.png");

\_newGameBtn.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

GameStats.restartGame(true);

GameStats.startGame();

BrieferScreen.getInstance().setVisible(false);

\_parent.setVisible(false);

}

});

add(\_newGameBtn);

}

/\*\*

\* Creates Back to menu button that closes the mapFrame and opens the menu

\*/

private void setUpBack() {

\_backToMenu = ImageTools.makeButton("srcImages\\Menu\\mainMenuIcon.png");

\_backToMenu.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

MainMenu.getInstance().setVisible(true);

MapGFrame.getInstance().setVisible(false);

BrieferScreen.getInstance().setVisible(false);

\_parent.setVisible(false);

}

});

add(\_backToMenu);

}

}