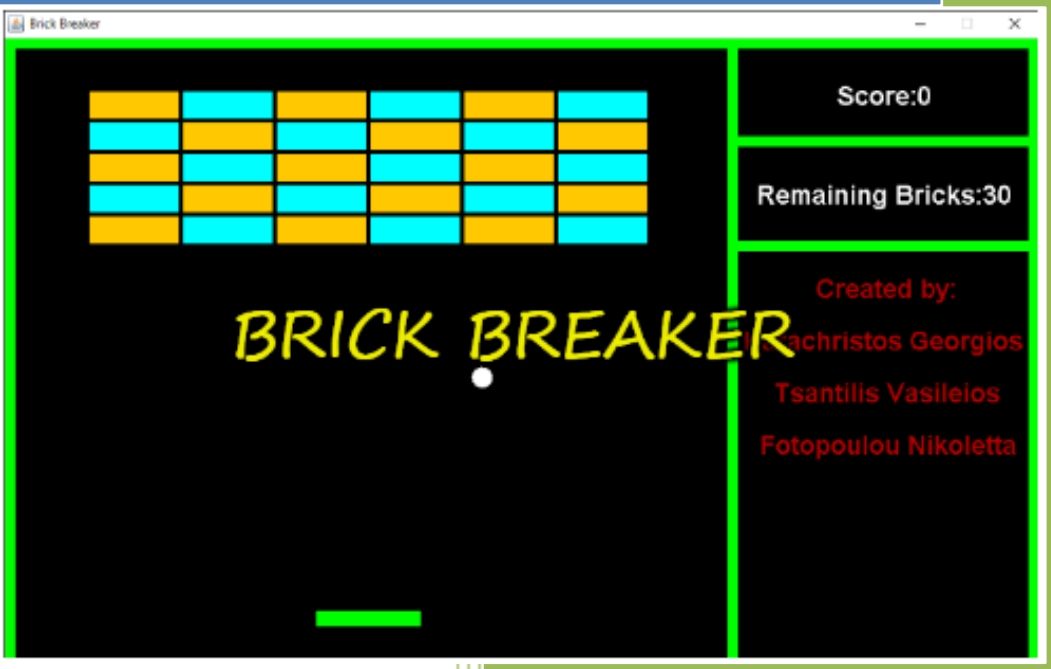
[2020]

Εργασία Αλληλεπίδραση Ανθρώπου-Μηχανής



Γιώργος Καραχρήστος (185192)

Βασίλης Τσαντίλης Ρέις (185296)

Νικολέττα Φωτοπούλου (185307)

Καθηγητής: Ευκλείδης Κεραμόπουλος

Περιεχόμενα

Εισαγωγή

Στα πλαίσια του μαθήματος Αλληλεπίδραση Ανθρώπου-Μηχανής επιλέξαμε να υλοποιήσουμε το παιχνίδι Brick Breaker σαν εργασία. Πρόκειται για ένα παιχνίδι φτιαγμένο το 1976, με στόχο ο παίκτης να σπάσει όλα τα τούβλα του παιχνιδιού χωρίς να αφήσει την μπάλα να πέσει κάτω, αλλά μετακινώντας κάθε φορά μία πλατφόρμα για να την σώσει.

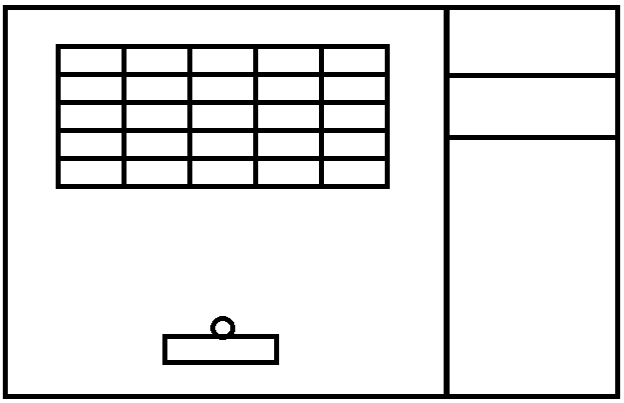
Επιλέξαμε το Brick Breaker μετά από μέρες αναζήτησης στα παιχνίδια που ήταν δυνατόν να φτιαχτούν με Java Swing, απορρίψαμε κάποιες ιδέες και καταλήξαμε σε κάποιο που ήταν περισσότερο στα μέτρα μας. Η μορφή του τελικού παιχνιδιού που φτιάξαμε ήταν εμπνευσμένη από το παιχνίδι Flipper και έτσι προστέθηκε στα δεξιά μια περιοχή στην οποία εμφανίζονται διάφορα στοιχεία του παιχνιδιού.

Η συνεργασία μας σαν ομάδα ήταν εξαιρετική, ο καθένας μπορούσε να επεξεργάζεται τον κώδικα και να προσθέτει ή να αφαιρεί κομμάτια, πάντα με την χρήση σχολίων για να εξηγείται το κάθε βήμα αναλυτικά.

Θα ακολουθήσει η παρουσίαση του πρωτότυπου και στην συνέχεια κάποια βασικά κομμάτια κώδικα με την ανάλυσή τους.

Περιγραφή πρωτοτύπου

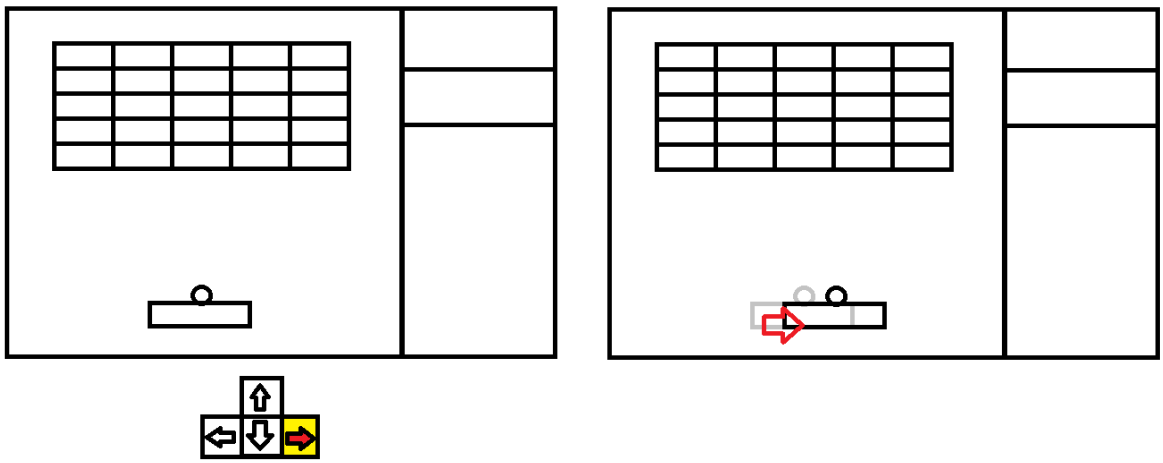
Μορφή εφαρμογής



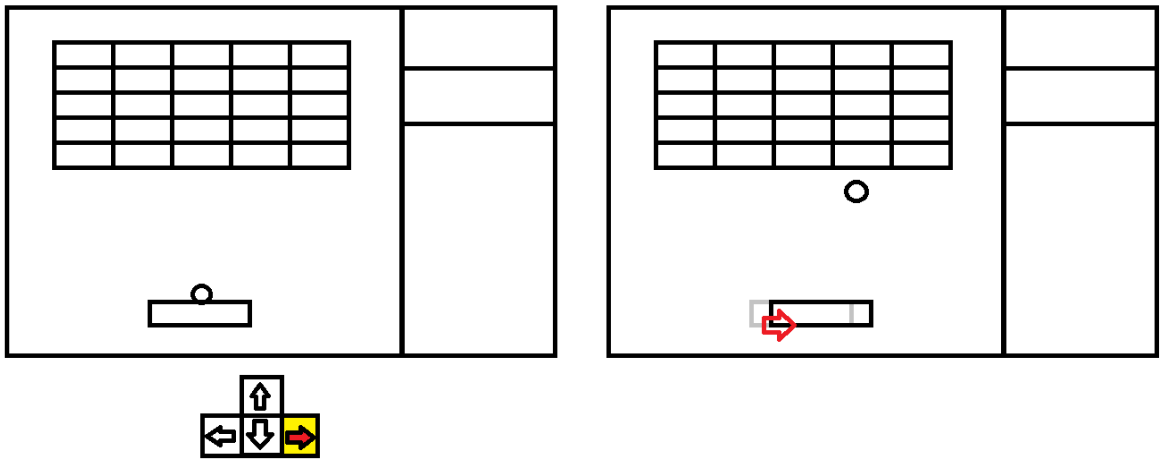
Η εφαρμογή έχει απλή μορφή που διευκολύνει τον χρήστη να εντοπίσει το βασικό παιχνίδι και να το ξεχωρίσει από το δεξί μέρος, το οποίο είναι μια περιοχή για να εμφανίζονται διάφορες πληροφορίες, όπως το σκορ, τα τούβλα που απομένουν και τα ονόματα των δημιουργών.

Ακόμα και στην χρήση του, είναι ιδιαίτερα απλή καθώς οι ενέργειες που πρέπει να κάνει ο χρήστης για να παίξει.

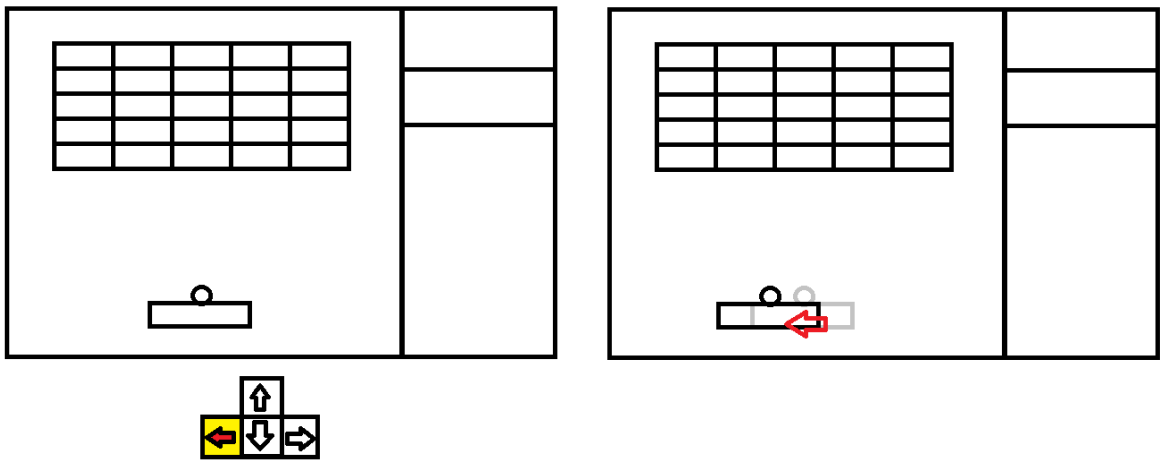
Λειτουργία



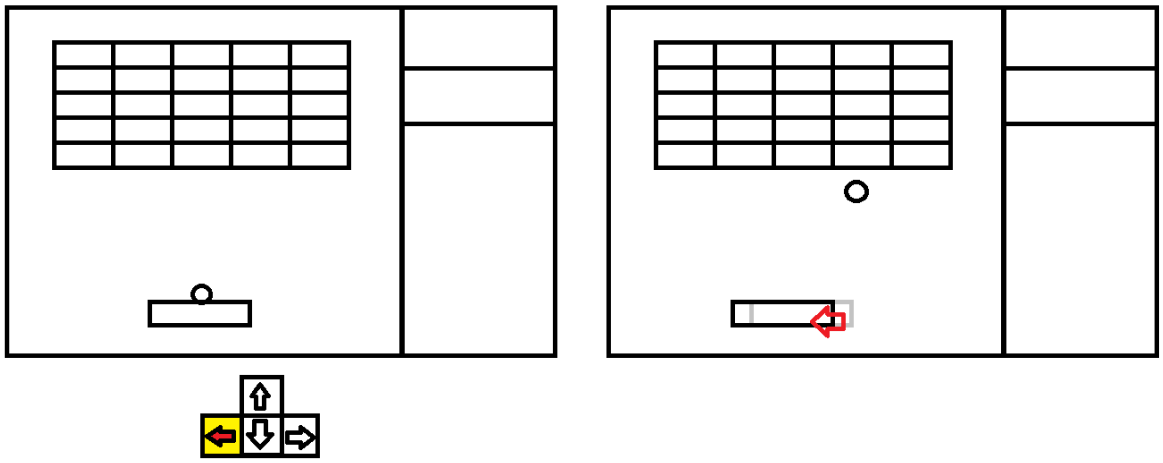
Με την έναρξη του παιχνιδιού, ο χρήστης έχει μπροστά του την πλατφόρμα μαζί με την μπάλα, τα τούβλα που πρέπει να σπάσει και το δεξί πλαίσιο πληροφοριών. Με το πάτημα του δεξιού πλήκτρου η πλατφόρμα μαζί με την μπάλα θα μετακινηθούν προς τα δεξιά.



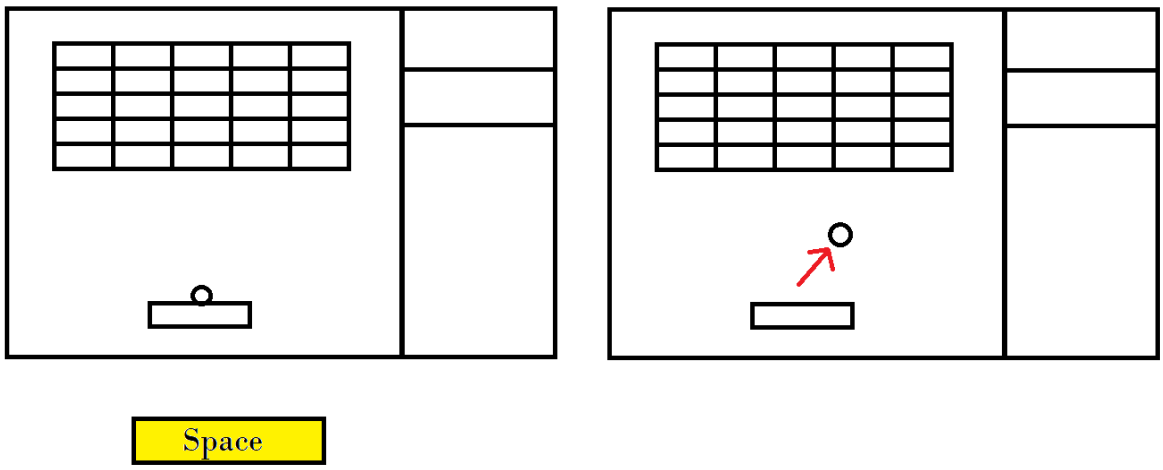
Φυσικά, κατά την διάρκεια του παιχνιδιού που η μπάλα κινείται ανεξάρτητα από τα βελάκια στον χώρο, το δεξί πλήκτρο θα μετακινήσει στα δεξιά μόνο την πλατφόρμα.



Αντίστοιχα, το πάτημα του αριστερού πλήκτρου θα έχει ως αποτέλεσμα την μετακίνηση της πλατφόρμας και της μπάλας προς τα αριστερά.

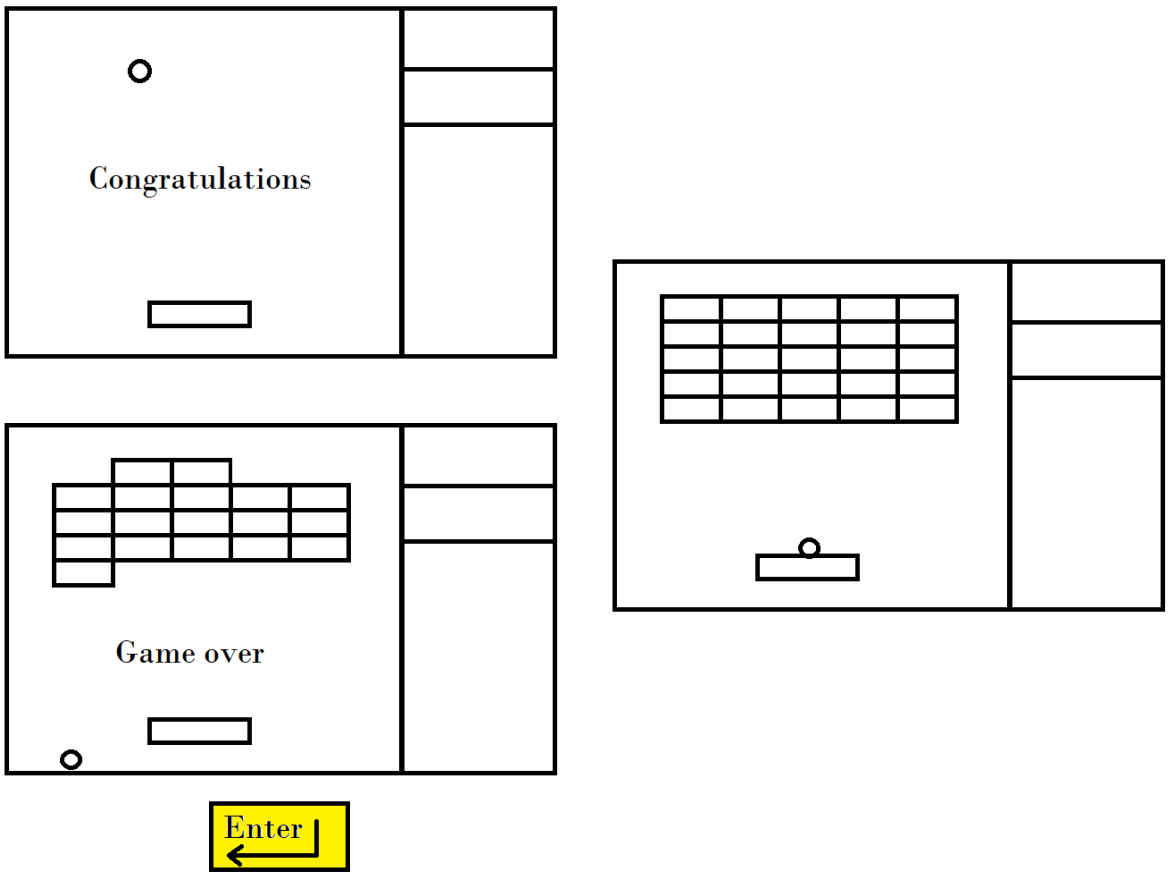


Όπως και πριν, εφόσον έχει ξεκινήσει το παιχνίδι, το πάτημα του αριστερού πλήκτρου θα μετακινήσει μόνο την πλατφόρμα στα αριστερά.



Με την έναρξη του παιχνιδιού, ο χρήστης έχει μπροστά του την πλατφόρμα μαζί με την μπάλα, τα τούβλα που πρέπει να σπάσει και το δεξί πλαίσιο πληροφοριών. Η μπάλα είναι τοποθετημένη πάνω στην πλατφόρμα και μετακινούνται μαζί. Η μπάλα απελευθερώνεται και το παιχνίδι ξεκινάει όταν πατηθεί το πλήκτρο Space.

Το πάτημα του Space θα έχει αποτέλεσμα μόνο πριν την εκκίνηση του παιχνιδιού.



Το παιχνίδι τελειώνει με δύο τρόπους: είτε ο χρήστης κερδίζει σπάζοντας όλα τα τούβλα, είτε χάνει καθώς του πέφτει κάτω η μπάλα.

Και στις δύο περιπτώσεις, αν πατηθεί το πλήκτρο Enter από το πληκτρολόγιο, το παιχνίδι ξεκινάει από την αρχή και όλα τοποθετούνται στην αρχική τους θέση.

Το πάτημα του Enter κατά την διάρκεια του παιχνιδιού δεν θα έχει κάποιο αποτέλεσμα.

Εφαρμογή-Κώδικας

Κλάση Game

Η κλάση Game είναι αυτή που περιέχει όλα τα βασικά στοιχεία του παιχνιδιού, από τη μορφή ως τις λειτουργίες. Οι τέσσερις ενέργειες που καλείται ο χρήστης να εκτελέσει (μετακίνηση πλατφόρμας δεξιά, αριστερά, έναρξη και επανέναρξη του παιχνιδιού) βρίσκονται επίσης σε αυτή τη κλάση.

Οι βασικές ενέργειες που πρέπει να εκτελέσει ο χρήστης έχουν προγραμματιστεί με τον ακόλουθο τρόπο:

Χρήση δεξιού και αριστερού βελών

```
209     public void keyPressed(KeyEvent e) {
210         if (e.getKeyCode()== KeyEvent.VK_RIGHT){
211             if(platformX+20>595)
212                 platformX=595;
213             else
214                 platformX+=20;
215             if (!play && totalBricks >0)
216                 ballX=platformX+40;
217         }
218         if(e.getKeyCode()==KeyEvent.VK_LEFT){
219             if(platformX-20<11)
220                 platformX=11;
221             else
222                 platformX-=20;
223             if (!play && totalBricks >0)
224                 ballX=platformX+40;
```

Μέσα στην μέθοδο keyPressed έχουμε όλες τις συνθήκες για το τι θα συμβεί εάν πατηθεί κάποιο πλήκτρο.

Αρχικά εάν πατηθεί το δεξί βελάκι, η λειτουργία του θα πρέπει να είναι να προχωρήσει προς τα δεξιά η πλατφόρμα ή η πλατφόρμα μαζί με την μπάλα αν είμαστε στην αρχή του παιχνιδιού. Στη γραμμή 210 ελέγχουμε αν η ενέργεια αυτή θα έχει ως αποτέλεσμα να ξεπεράσει η πλατφόρμα τα όρια που έχουμε βάλει, για αυτό αν ισχύει την επαναφέρουμε στην προηγούμενη θέση, αλλιώς προχωράει κανονικά 20 πίξελ δεξιά.

Στη γραμμή 214 ελέγχουμε άμα βρισκόμαστε στην αρχή του παιχνιδιού, άρα το παιχνίδι δεν τρέχει (play==false→!play==true) και άμα υπάρχουν τούβλα στο παιχνίδι. Εάν ισχύουν αυτά, η μπάλα

θα μετακινηθεί και αυτή 20 πίξελ ή αλλιώς θα βρίσκεται 40 πίξελ μετά την τοποθεσία της πλατφόρμας (στον άξονα x), ώστε να είναι τοποθετημένη πάντα στο κέντρο της.

Με παρόμοιο τρόπο στο if για το αν πατηθεί το αριστερό βελάκι, ελέγχουμε αν η πλατφόρμα θα ξεπεράσει τα αριστερά όρια, και αν ναι την επαναφέρουμε στην προηγούμενη θέση της, αλλιώς κινείται 20 πίξελ αριστερά. Στην συνέχεια, αν το παιχνίδι δεν τρέχει και αν υπάρχουν τούβλα στο παιχνίδι, η μπάλα θα μετακινηθεί επίσης και θα βρίσκεται 40 πίξελ μετά τη θέση της πλατφόρμας (στον άξονα x) και θα βρίσκεται στο κέντρο της.

Τα 40 πίξελ υπολογίσθηκαν από το πλάτος της πλατφόρμας/2 μείων το πλάτος της μπάλας/2.

Χρήση Space και Enter

```
226             if(e.getKeyCode()==KeyEvent.VK_SPACE ){
227                 play=true;
228             }
229             if(e.getKeyCode()==KeyEvent.VK_ENTER ){
230                 if(!play && totalBricks ==0 || ballY > 650){
231                     ballColor=Color.WHITE;
232                     ballX=340;
233                     ballY=530;
234                     ballVelocityX=1;
235                     ballVelocityY=2;
236                     platformX=300;
237                     score=0;
238                     totalBricks=30;
239                     theBricks= new Bricks(5,6);
240                     rightColor=Color.GREEN;
241                     topColor=Color.GREEN;
242                     leftColor=Color.GREEN;
243
244                     repaint();
245                 }
```

Η ενέργεια του space είναι μόνο μία, να ξεκινήσει το παιχνίδι. Έτσι με το πάτημά του η μεταβλητή play που καθορίζει την κατάσταση του παιχνιδιού γίνεται true.

Το enter χρησιμοποιείται σε δύο καταστάσεις: είτε ο χρήστης έχει κερδίσει άρα το Play==false και τα συνολικά τούβλα είναι ίσα με το μηδέν, ή έχει χάσει καθώς η μπάλα έχει πέσει κάτω (έχει ξεπεράσει ένα συγκεκριμένο όριο στον άξονα y).

Και στις δύο περιπτώσεις με το enter το παιχνίδι επαναφέρεται στην αρχική του κατάσταση, οπότε αρχικοποιούμε τις μεταβλητές που χρησιμοποιήσαμε και αλλάζαμε κατά τη διάρκεια του παιχνιδιού: το χρώμα της μπάλας γίνεται άσπρο, η μπάλα τοποθετείται στο κέντρο της πλατφόρμας η οποία βρίσκεται στο κέντρο του παιχνιδιού, η μπάλα θα ξεκινήσει να κινείται κατά ένα πίξελ στον άξονα x και 2 στον y, μηδενίζεται το σκορ, τα τούβλα στο πλήθος είναι πάλι 30 και ορίζονται από την κλάση Bricks με 5 γραμμές και 6 στήλες και τα τρία όρια που άλλαζαν χρώμα επαναφέρονται στο αρχικό τους χρώμα, το πράσινο.

Τέλος, με την repaint ζωγραφίζονται ξανά όσα βρίσκονται στην κλάση paint που είναι: τα υπόλοιπα όρια, η μέθοδος για να ζωγραφιστούν τα τούβλα, τα στοιχεία για το δεξί μέρος που είναι το σκορ, τα τούβλα που μένουν, τα ονόματα των δημιουργών και διάφορες συνθήκες για τα μηνύματα που θα εμφανίζονται ανάλογα με τις ενέργειες του χρήστη.

Πολύ σημαντικά είναι όσα συμβαίνουν κατά την διάρκεια του παιχνιδιού:

Action performed

```
123     public void actionPerformed(ActionEvent e) {
124         timer.start();
125         if(play){
126             if( new Rectangle(ballX,ballY,20,20).intersects(new Rectangle(platformX,550,100,15))){
127                 if(ballY>535)
128                     ballVelocityX= -ballVelocityX;
129                 else
130                     ballVelocityY= -ballVelocityY;
            }
```

Στην Action Performed έχουμε όλα όσα θα συμβούν όσο τρέχει το παιχνίδι και το Play είναι true.

Αρχικά στην γραμμή 125 ελέγχουμε τι θα γίνει στην περίπτωση που η μπάλα αγγίζει την πλατφόρμα. Για να το κάνουμε αυτό δημιουργούμε ένα αόρατο ορθογώνιο παραλληλόγραμμο γύρω από την μπάλα με την διαστάσεις της μπάλας, και ένα άλλο ίδιο με τις διαστάσεις της πλατφόρμας. Έτσι σε αυτή τη συνθήκη ελέγχουμε εάν αυτά τα δύο αόρατα παραλληλόγραμμα συγκρούονται.

Σε αυτό το σημείο θέλουμε αν η μπάλα ακουμπήσει την πλατφόρμα από την πάνω πλευρά να αλλάξει το y της, δηλαδή να ξανανέβει πάνω. Όμως, θέλουμε αν χτυπήσει στα πλάγια την πλατφόρμα να αλλάζει μόνο το x και να πέφτει κάτω. Στην γραμμή 126 ελέγχουμε σε ποια από τις δύο περιπτώσεις βρισκόμαστε την στιγμή που γίνεται η σύγκρουση και αλλάζει είτε το x είτε το y.

```
131                 LOOP:
132                 for (i=0; i<theBricks.TheBricks.length; i++) {
133                     for (j = 0;j<theBricks.TheBricks[0].length;j++) {
134                         if (theBricks.TheBricks[i][j]>0) {
135                             brickX= j * theBricks.brickWidth + 80;
136                             brickY= i * theBricks.brickHeight + 50;
137                             brickWidth=theBricks.brickWidth;
138                             brickHeight=theBricks.brickHeight;
139                             Rectangle brickRect = new Rectangle(brickX,brickY,brickWidth,brickHeight);
140                             Rectangle ballRect=new Rectangle(ballX,ballY,20,20);
141                             if (ballRect.intersects(brickRect)) {
142                                 theBricks.setBrickValue(0,i,j);
143                                 totalBricks--;
144                                 score +=5;
145                                 if (i==j || i+2==j || i+4==j || i-2==j || i-4==j)
146                                     ballColor=Color.orange;
147                                 else
148                                     ballColor=Color.cyan;
149                                 if (ballX + 19 <=brickRect.x || ballX+1 >=brickRect.x + brickRect.width)
150                                     ballVelocityX = -ballVelocityX;
151                                 else
152                                     ballVelocityY = -ballVelocityY;
153                                 break LOOP;
154                             }
155                         }
156                     }
157                 }
```

Μπαίνουμε σε ένα loop με επαναλήψεις όσες και ο αριθμός των τούβλων που υπάρχουν εκείνη τη στιγμή. Στην κλάση Bricks έχουμε ορίσει έναν πίνακα με 5 γραμμές και 6 στήλες και το περιεχόμενο κάθε κελιού είναι 1 (θα δούμε ότι όσα έχουν την τιμή 1 ζωγραφίζονται ξανά).

Ελέγχουμε κάθε φορά αν το περιεχόμενο του κάθε κελιού είναι μεγαλύτερο του 0, δηλαδή 1, άρα δεν έχει σπάσει ακόμα από την μπάλα.

Σκοπός μας τώρα είναι να δημιουργήσουμε γύρω από κάθε τούβλο ένα ορθογώνιο, το οποίο είναι αυτό όπου θα ανιχνευτεί στην συνέχεια στη σύγκρουση του με την μπάλα. Για να το τοποθετήσουμε εκεί που είναι το συγκεκριμένο τούβλο χρειαζόμαστε το x, το y και τις διαστάσεις του.

Οι τύποι του x και του y στις γραμμές 135-136 θα εξηγηθούν αναλυτικά στην κλάση Bricks όπου δημιουργούνται τα τούβλα, ενώ στις δύο επόμενες σειρές παίρνουμε από την κλάση Bricks το πλάτος και το ύψος των τούβλων και τα αποθηκεύουμε σε μεταβλητές. Έτσι δημιουργούμε για κάθε τούβλο ένα αόρατο ορθογώνιο με τις διαστάσεις του.

Στη συνέχεια δημιουργώντας ένα αόρατο πάλι ορθογώνιο γύρω από την μπάλα με τις διαστάσεις της, ελέγχουμε εάν συγκρούεται με το ορθογώνιο του κάθε τούβλου. Αν ναι, μηδενίζουμε την τιμή αυτού του κελιού, μειώνουμε κατά ένα τον αριθμό των τούβλων, αυξάνουμε κατά 5 το σκορ.

Στις γραμμές 145-148 βρίσκεται η συνθήκη για να αλλάζει χρώμα η μπάλα ανάλογα με το τούβλο που χτυπάει. Έπειτα, πρέπει να δημιουργήσουμε την συνθήκη που θα καθορίζει την πορεία της μπάλας ανάλογα με το ποια πλευρά του τούβλου (ή αλλιώς του ορθογωνίου που δημιουργήσαμε) χτυπάει.

Θέλουμε αν χτυπάει από δεξιά ή αριστερά να αλλάζει το x, ενώ από κάτω ή πάνω να αλλάζει το y. Έτσι στη συνθήκη που βρίσκεται γραμμή 149 του λέμε αν η δεξιά πλευρά της μπάλας (το x της +19 αφού έχει 20 πίξελ διάμετρο) είναι μικρότερη ή ίση του x του τούβλου (εκεί που είναι η αριστερή

πλευρά του δηλαδή) ή αν το x της μπάλας, η αριστερή πλευρά της δηλαδή +1 είναι μεγαλύτερη ή ίση με την τοποθεσία της δεξιάς πλευράς του τούβλου (το x του τούβλου +το πλάτος του). Βέβαια, χρησιμοποιούμε σε όλη αυτή τη διαδικασία τις συντεταγμένες και τις διαστάσεις του ορθογωνίου που έχουμε δημιουργήσει. Έτσι, αν ισχύει κάτι από τα παραπάνω αλλάζει το x της μπάλας, αλλιώς (δηλαδή αν χτυπήσει η μπάλα από πάνω ή από κάτω το τούβλο) αλλάζει το y.

Όταν μιλάμε για αλλαγές στο x και y της μπάλας, αναφερόμαστε στην κατεύθυνση της, και αυτή η αλλαγή γίνεται βάζοντας μπροστά τους το μείων. Στην περίπτωση λοιπόν που η μπάλα χτυπούσε δύο τούβλα ταυτόχρονα, παίρνοντας 2 φορές το μείων συνέχιζε την πορεία της κανονικά έχοντας εξαφανίσει δύο τούβλα την ίδια στιγμή. Για αυτό προστέθηκαν το LOOP και break LOOP, έτσι ώστε στην πρώτη αλλαγή φοράς της μπάλας να απομακρύνεται στην αντίθετη κατεύθυνση χωρίς να σπάσει το δεύτερο τούβλο και βγαίνει εντελώς έξω από το LOOP.

```
158                 ballX += ballVelocityX;
159                 ballY += ballVelocityY;
160                 if(ballX <10){
161                     ballVelocityX = -ballVelocityX;
162                     if(leftColor==Color.green)
163                         leftColor=Color.yellow;
164                     else if(leftColor==Color.yellow)
165                         leftColor=Color.cyan;
166                     else if(leftColor==Color.cyan)
167                         leftColor=Color.green;}
168                 if(ballY < 10){
169                     ballVelocityY = -ballVelocityY;
170                     if(topColor==Color.green)
171                         topColor=Color.yellow;
172                     else if(topColor==Color.yellow)
173                         topColor=Color.cyan;
174                     else if(topColor==Color.cyan)
175                         topColor=Color.green;}
176                 if(ballX >675){
177                     ballVelocityX = -ballVelocityX;
178                     if(rightColor==Color.green)
179                         rightColor=Color.yellow;
180                     else if(rightColor==Color.yellow)
181                         rightColor=Color.cyan;
182                     else if(rightColor==Color.cyan)
183                         rightColor=Color.green;}
184             }
185             repaint();
```

Βγαίνουμε λοιπόν από τα for και συνεχίζουμε στο ότι εφόσον το παιχνίδι είναι true, η μπάλα πρέπει να κινείται στον άξονα x όσα πίξελ έχουμε ορίσει το ballVelocityX να είναι, ενώ στον άξονα y όσα πίξελ έχουμε ορίσει το ballVelocityY να είναι.

Μετά έχουμε τις τρεις συνθήκες για τα τρία όρια αριστερά, πάνω και δεξιά και σε κάθε περίπτωση αλλάζουμε την πορεία της μπάλας στον άξονα x ή y. Επιπλέον, σε κάθε μία από τις συνθήκες έχουμε προσθέσει κάποιες γραμμές ώστε ακουμπώντας η μπάλα το όριο να το αλλάζει κάθε φορά χρώμα, ανάλογα με το ποιο ήταν το προηγούμενο.

Κλάση Bricks

Μέθοδος draw

```
25      public void draw(Graphics2D graph) {
26          for(i=0;i<TheBricks.length; i++){
27              for(j=0; j < TheBricks[0].length; j++ ){
28                  if(i==j || i+2==j || i+4==j || i-2==j || i-4==j)
29                      brickColor=Color.orange;
30                  else
31                      brickColor=Color.cyan;
32                  if (TheBricks[i][j]>0){
33                      graph.setColor(brickColor);
34                      graph.fillRect(j*brickWidth+80,i*brickHeight+50,brickWidth,brickHeight);
35                      graph.setStroke(new BasicStroke(4));
36                      graph.setColor(Color.black);
37                      graph.drawRect(j*brickWidth+80,i*brickHeight+50,brickWidth,brickHeight);
38                  }
39              }
40          }
41      }
```

Η μέθοδος draw στην κλάση Bricks είναι αυτή που θα ζωγραφίσει τα τούβλα με γραφικά 2D.

TheBricks είναι ένας πίνακας τύπου Bricks που δημιουργείται μέσα στην κλάση με κελιά όσα και τα τούβλα που θέλουμε να δημιουργήσουμε. Έτσι στην draw κάνουμε επαναλήψεις όσες και τα κελιά. Γραμμή 28-31 έχουμε την συνθήκη για να χρωματίσει τα τούβλα.

Στην 32 ελέγχουμε αν η τιμή του κελιού είναι μεγαλύτερη του 0, άρα το τούβλο πρέπει να ζωγραφιστεί. Παίρνει χρώμα ανάλογα με την πάνω συνθήκη και η συντεταγμένες του ορίζονται ως εξής:

Για να βρίσκεται όλο το σύνολο των τούβλων κεντρικά μέσα στο παιχνίδι και να είναι ομοιόμορφο, αφήνουμε ένα κενό διάστημα 50 πίξελ από πάνω και 80 πίξελ από δεξιά και αριστερά. Επομένως το κάθε τούβλο που θα δημιουργείται πρέπει να μετακινείται 80 πίξελ δεξιά και 50 κάτω.

Ταυτόχρονα όμως, το κάθε τούβλο θα πρέπει να προσπερνάει τα προηγούμενα τούβλα για να φτάσει στη θέση του, και αυτό επιτυγχάνεται πολλαπλασιάζοντας τις διαστάσεις του με το i και το j.

Έτσι, το πρώτο τούβλο θα έχει για x =0*brickWidth+80, δηλαδή απλώς 80 πίξελ δεξιά, ενώ το δεύτερο 1*brickWidth+80, δηλαδή αρχικά όσο είναι το πλάτος ενός τούβλου (για να προσπεράσει το προηγούμενο) +80 πίξελ που είναι το προκαθορισμένο κενό στα δεξιά. Η αντίστοιχη διαδικασία ακολουθείται με το y για να κατέβουν προς τα κάτω τα τούβλα.

Καλούμε την fillRect με τα ανάλογα x,y,πλάτος και ύψος για να δημιουργήσουμε περιγράμματα γύρω από τα τούβλα και έπειτα την BasicStroke για να δώσουμε πάχος 4 πίξελ ώστε να διαχωρίζονται μεταξύ τους τα τούβλα.

Επίλογος

Στην περίοδο των δύο μηνών που ασχολούμαστε με αυτή την εργασία αποκτήσαμε επιπλέον γνώσεις πάνω στον τομέα της πληροφορικής και ζήσαμε μια εμπειρία που ανέπτυξε τις συνεργατικές ικανότητες μας. Τέλος, αποκομίσαμε δεξιότητες που θα μας φανούν χρήσιμες σε μελλοντικές εργασίες.