```java
import processing.core.PApplet;
import processing.core.PImage;

import java.util.ArrayList;

public class Boat {
    //All the boat's variables
    PApplet p;
    //The positions
    float xPos, yPos;
    //players money
    float  money;
    //boats sprite / appearance
    PImage boatPic;
   /*list of what boat have in its inventory of
   items and been item tells how much it has of that item*/
    ArrayList<Item> inventory;
    //The list that the boat can see
    ArrayList<Tile> showneTileSet = new ArrayList<Tile>();

    //---------- DEFAULT CONSTRUCTOR :) ----------\\
    public Boat(PApplet p, float x, float y, float m) {
        this.p = p;
        this.inventory = inventory;
        xPos = x;
        yPos = y;
        money = m;
    }
    //----------METHODS----------\\

    //This feature makes the boat inventory and fills it up with all the game items
    void generateInventory() {
        ArrayList StockInventory = new ArrayList<Item>();
        Item Banana = new Item(10, 0, "Banana");
        Item Rum = new Item(5, 0, "Rum");
        Item Eyepatch = new Item(3, 0, "Eyepatch");
        StockInventory.add(Banana);
        StockInventory.add(Rum);
        StockInventory.add(Eyepatch);
        inventory = StockInventory;
    }

    //This feature draws the boat
    void displayBoat(float pX, float pY, int s) {
        if (boatPic != null)
            p.image(boatPic, s * pX, s * pY, s, s);

```

```
48
49          for (int i = 0; i < showneTileSet.size(); ++i) {
50              Tile tile = showneTileSet.get(i);
51              if (xPos == tile.xPos && yPos == tile.yPos) {
52                  if (!tile.Contents.equals("WATER")) {
53                      moveBoatToWater();
54                  }
55
56              }
57          }
58      }
59
60      //Since the boat can not show it is on land and the boat is
61      // placed randomly on the map then make this
62      // function that the host boat moves to a water tile
63      void moveBoatToWater() {
64          boolean didFindWaterTile = false;
65
66          //checks if there is a water tile in the shovn tiles
67          for (int i = 0; i < showneTileSet.size(); ++i) {
68              Tile tile = showneTileSet.get(i);
69
70              //show there is back there and stop  the looped
71              if (tile.Contents.equals("WATER")) {
72                  xPos = tile.yPos;
73                  yPos = tile.yPos;
74                  didFindWaterTile = true;
75                  break;
76              }
77
78
79          }
80
81          //if no back water is found at random and check again
82          if (!didFindWaterTile) {
83              int ran = (int) p.random(0, showneTileSet.size());
84              xPos = showneTileSet.get(ran).xPos;
85              yPos = showneTileSet.get(ran).yPos;
86              moveBoatToWater();
87          }
88
89      }
90
91      //This feature fills the list with tiles that the boat can see
92      void fillUpShownTiles(ArrayList<Tile> tileSet) {
93          showneTileSet.clear();
94
```

```
 95
 96          for (int i = 0; i < tileSet.size(); ++i) {
 97              if (tileSet.get(i).xPos - xPos > -3 && tileSet.get(i).xPos - xPos < 3) {
 98
 99                  if (tileSet.get(i).yPos - yPos > -3 && tileSet.get(i).yPos - yPos < 3
     ) {
100
101                      showneTileSet.add(tileSet.get(i));
102                  }
103              }
104
105          }
106      }
107
108      //This feature provides a float with players' scores. by taking number of
     money and items nomal pric * with number
109      float calRating(){
110          float price = 0;
111          price += money;
112          for(int i = 0; i<inventory.size();++i){
113
114              price+=inventory.get(i).value * inventory.get(i).ammount;
115          }
116          return price;
117
118      }
119  }
120
```

```
 1  import processing.core.PApplet;
 2
 3  public class Item {
 4      //Items variables
 5      PApplet p;
 6      //items normal price and its current price
 7      float value, stockValue;
 8      //the name of the variable
 9      String Name;
10      //the number of the thing
11      int ammount;
12
13      //----------CONSTRUCTOR----------\\
14      Item(float v, int a, String n) {
15          value = v;
16          ammount = a;
17          Name = n;
18
19          stockValue = value;
20
21
22      }
23
24  }
25
```

```
 1
 2  import processing.core.PApplet;
 3  import processing.core.PImage;
 4  import processing.data.Table;
 5  import javax.sound.sampled.*;
 6  import java.io.File;
 7  import java.util.Objects;
 8  import java.util.Scanner;
 9  import javax.sound.sampled.AudioInputStream;
10  import javax.sound.sampled.AudioSystem;
11  import javax.sound.sampled.Clip;
12  import java.io.File;
13
14  public class main extends PApplet {
15      public static void main(String[] args) {
16
17          PApplet.main("main");
18      }
19      //background music
20      public static Clip bgmusic;
21      //all the menus
22      public PauseMenu pauseMenu;
23      public SettingMenu settingMenu;
24      public static MainMenu mainMenu;
25      //The play the board
26      public GameBoard gb;
27      //background image
28      public static PImage bg;
29
30      @Override
31      public void settings() {
32          //sets the size of the window
33          size(1280, 720);
34      }
35
36
37      //sets up the game by loading music, pictures and constructing all the
    classes
38      @Override
39      public void setup() {
40          bg = loadImage("Tegnebraet 2.png");
41          settingMenu = new SettingMenu(this);
42          mainMenu = new MainMenu(this);
43          pauseMenu = new PauseMenu(this, settingMenu, mainMenu);
44          gb = new GameBoard(this, pauseMenu);
45
46          textSize(12);
```

```
47          text("Loading", width / 2 - textWidth("Loading") / 2, height / 2);
48          try {
49              File file = new File("Music/bg.wav");
50              Scanner scanner = new Scanner(System.in);
51              AudioInputStream audioInputStream = AudioSystem.
    getAudioInputStream(file);
52              bgmusic = AudioSystem.getClip();
53              bgmusic.open(audioInputStream);
54              bgmusic.loop(Clip.LOOP_CONTINUOUSLY);
55
56
57
58
59          } catch (Exception e) {
60              e.printStackTrace();
61          }
62
63
64
65
66
67      }
68
69
70
71      //loaded mappt from csv file vilel put it in savemanger but it pde could not
    figure out
72      public void loadedMap(File selection) {
73          if (selection == null) {
74              println("Window was closed or the user hit cancel.");
75          } else {
76              Table map = null;
77
78              try {
79                  map = loadTable(selection.getAbsolutePath());
80                  println("User selected " + selection.getAbsolutePath() + " First tile
    : " + map.getString(0, 0));
81                  gb.saveManger.loadGame(32, map, gb.tileSet);
82              } catch (NullPointerException ignored) {
83                  System.out.println(selection.getAbsolutePath());
84              }
85          }
86      }
87
88      //This function is suitable for anything on the window
89      @Override
90      public void draw() {
```

```
 91
 92          clear();
 93          background(200);
 94
 95          if (settingMenu.visible) {
 96              settingMenu.drawMenu();
 97          }
 98          if (gb.visible) {
 99              gb.drawBoard();
100          }
101          if (pauseMenu.visible) {
102              pauseMenu.drawMenu();
103          }
104
105          if (mainMenu.visible) {
106              mainMenu.drawMenu();
107
108          }
109
110
111      }
112
113      //Do this function when the mouse is clicked
114      @Override
115      public void mouseClicked() {
116
117          if (settingMenu.visible) {
118
119              settingMenu.menuMouseClick();
120              settingMenu.btnChangeScreen(mainMenu, gb);
121          }
122          if (gb.visible) {
123              gb.boardmouseClicked();
124          }
125
126          if (mainMenu.visible) {
127              mainMenu.menuMouseClick(mouseX, mouseY);
128          }
129
130          if (pauseMenu.visible) {
131              pauseMenu.clickMenu(mouseX, mouseY);
132          }
133
134      }
135
136      //This function runs when the keyboard is clicked
137      @Override
```

```
138     public void keyPressed() {
139
140         if (key == ESC) {
141             key = 0;
142             if (gb.pauseMenu.visible) {
143                 gb.pauseMenu.aktiverGameBord();
144             } else if (gb.visible) {
145                 gb.aktiverPauseMenu();
146             }
147
148         }
149
150         if (keyCode == 0) {
151             gb.akitverDevconsole();
152         }
153
154     }
155
156     //This function runs when the keyboard is clicked
157     public void keyTyped() {
158
159         settingMenu.menuKeyTyped();
160         gb.keyTyped();
161     }
162
163
164 }
165
```

```
 1  import processing.core.PApplet;
 2  import processing.core.PImage;
 3
 4  public abstract class Tile {
 5      PApplet p;
 6
 7      //This variable determines how large the line of tiles should be
 8      float strong = 0;
 9      //This variable will be used for path finding
10      int path = 10000;
11      //the image of the image
12      PImage tileImage;
13      //This boolean tells you if the tile is pressed
14      Boolean selectedTile = false;
15      boolean cliked = false;
16      //This string tells what the tilen contains
17      String Contents;
18      //the positions
19      int xPos;
20      int yPos;
21
22      //----------CONSTRUCTOR----------
23      public Tile(PApplet p, String C, float x, float y) {
24          Contents = C;
25          this.p = p;
26          xPos = (int) x;
27          yPos = (int) y;
28      }
29      //----------METHODS----------\\
30
31      //This function checks tile clicked
32      void checkIfCliked(int pX, int pY, int s) {
33          int positionX = 0 + s * pX;
34          int positionY = 0 + s * pY;
35          if (p.mouseX > positionX &&
36                  p.mouseX < positionX + s &&
37                  p.mouseY > positionY &&
38                  p.mouseY < positionY + s) {
39              cliked = true;
40
41          }
42      }
43
44      //check show the mouse is over it
45      void checkIfMouseOver(float pX, float pY, int s) {
46          int positionX = (int) (0 + s * pX);
47          int positionY = (int) (0 + s * pY);
```

```
48          if (p.mouseX > positionX &&
49                  p.mouseX < positionX + s &&
50                  p.mouseY > positionY &&
51                  p.mouseY < positionY + s) {
52              p.fill(200, 200, 200, 200);
53              p.strokeWeight(10);
54              p.stroke(strong * 255, strong * 190, 0);
55              p.rect(0 + s * pX, 0 + s * pY, s, s);
56
57          }
58      }
59
60      //draws it
61      void Display(float pX, float pY, int s, Boolean pic, float strokeSize) {
62
63          PImage tilephoto = null;
64          if (Contents.equals("WATER")) {
65              p.fill(60, 100, 200);
66
67          } else if (Contents.equals("SAND")) {
68              p.fill(190, 181, 115);
69
70          } else {
71
72              if (Contents.equals("GRASS")) {
73                  p.fill(84, 201, 71);
74              } else if (Contents.equals("SHOP")) {
75                  p.fill(201, 0, 191);
76              } else {
77                  p.fill(200);
78              }
79          }
80          if (tileImage != null) {
81              if (pic)
82                  tilephoto = tileImage;
83          }
84
85          p.strokeWeight(strokeSize * 2);
86          p.stroke((strong * 255), strong * 190, 0);
87          p.rect(s * pX, s * pY, s, s);
88          if (tilephoto != null && pic) {
89              p.image(tilephoto, 0 + s * pX, 0 + s * pY, s, s);
90              p.fill(0);
91              String tileInfo = xPos + "x" + yPos;
92              p.text(tileInfo, s * pX + s / 2 - p.textWidth(tileInfo) / 2, s * pY + s / 2);
93          }
94
```

```
 95
 96          if (selectedTile) {
 97              p.fill(200, 200, 200, 200);
 98              p.strokeWeight(10);
 99              p.stroke(strong * 255, strong * 190, 0);
100              p.rect(0 + s * pX, 0 + s * pY, s, s);
101
102          }
103      }
104
105      //used for shoptiles
106      public abstract void drawShopMenu(Player player, float scaleSize);
107      public abstract void showShop(float s);
108      public abstract void clickShop();
109      public abstract void setTileImage();
110  }
111
```

```
1  import processing.core.PApplet;
2
3  public abstract class Button {
4      //button variables
5      PApplet p;
6      //button "positions" and "size".
7      float positionX, positionY, sizeX, sizeY;
8      //button size and position on the screen
9      float realPositionX, realPositionY, realSizeX, realSizeY;
10
11     //The size. This size is multiplied by "positions" and "size"
12     // to get the positions and size of the screen.
13     // that way the game can font window size without destroying anything
14     float size = 1;
15
16     //mouse x and y postion
17     float mouseX, mouseY;
18
19     //The string with infoamtion inside the button
20     String text;
21
22     //boolean if the button is clicked
23     boolean clicked = false;
24
25     //---------- CONSTRUCTOR :) ----------\\
26     Button(PApplet papp, int posX, int posY, int sizeX, int sizeY, String text) {
27         p = papp;
28         positionX = posX;
29         positionY = posY;
30         this.sizeX = sizeX;
31         this.sizeY = sizeY;
32         this.text = text;
33         realSizeY = sizeY;
34         realSizeX = sizeX;
35
36         realPositionY = posY;
37         realPositionX = posX;
38
39     }
40
41     //----------METHODS----------\\
42
43
44     //draws the button
45     void drawButton() {
46         positionX = realPositionX * size;
47         positionY = realPositionY * size;
```

```
48          sizeX = realSizeX * size;
49          sizeY = realSizeY * size;
50
51
52          p.stroke(1, 46, 74, 100);
53          if (clicked) {
54              p.fill(255);
55          } else {
56              p.fill(200, 200, 200, 100);
57          }
58
59          p.rect(positionX, positionY, sizeX, sizeY);
60          p.fill(0);
61          p.textSize(16 * size);
62          p.text(text, positionX + (sizeX / 16), positionY + (sizeY / 2));
63      }
64
65
66
67      //return if the button is clicked
68      boolean isClicked() {
69          return clicked;
70      }
71
72      //The one who puts about the click is true
73      abstract void registerClick(float mouseX, float mouseY);
74  }
75
```

```
1  import processing.core.PApplet;
2  import processing.core.PVector;
3
4  import java.util.ArrayList;
5
6  public class Player<oldpath> extends Boat {
7
8      //This lsite is filled with the players' chosen route
9      ArrayList<PVector> path = new ArrayList<>();
10
11     //---------- CONSTRUCTOR :) ----------\\
12     public Player(PApplet p, float x, float y,  float m) {
13
14         super(p, x, y,  m);
15         path.add(null);
16
17     }
18     //----------METHODS----------\\
19
20     //find the best tile to go to based on 2 tiles
21     int neighBordPath(ArrayList<Tile> tileSet, int chosen, int extra, int path) {
22         int re = Math.min(path, tileSet.get(chosen + extra).path);
23         return re;
24     }
25
26     //This function gives all tiles from the selected
27     // tile a number of how far it is from the selected tile
28     // and forges the fastest route to the selected tile and adds the items to the
   list path
29     void findPath(Tile selected, ArrayList<Tile> tileSet) {
30         int playerTileNum = 0;
31         selected.path = 0;
32
33         boolean pathAdded = false;
34         do {
35             pathAdded = false;
36             for (int i = 0; i < tileSet.size(); ++i) {
37                 Tile tile = tileSet.get(i);
38
39                 if (!tile.Contents.equals("WATER")) {
40                     tile.path = Integer.MAX_VALUE;
41                     continue;
42                 }
43
44                 if (selected == tile)
45                     continue;
46
```

```
47              // If player positioned on tile
48              if (tile.xPos == xPos && tile.yPos == yPos)
49                  playerTileNum = i;
50
51              int pathNumber = 10000;
52              pathNumber = neighBordPath(tileSet, i, +1, pathNumber);
53              pathNumber = neighBordPath(tileSet, i, +35, pathNumber);
54              pathNumber = neighBordPath(tileSet, i, +36, pathNumber);
55              pathNumber = neighBordPath(tileSet, i, +37, pathNumber);
56              pathNumber = neighBordPath(tileSet, i, -35, pathNumber);
57              pathNumber = neighBordPath(tileSet, i, -36, pathNumber);
58              pathNumber = neighBordPath(tileSet, i, -37, pathNumber);
59              pathNumber = neighBordPath(tileSet, i, -1, pathNumber);
60
61              // System.out.println("Path number: " + pathNumber);
62
63              if (tile.path > pathNumber + 1) {
64                  tile.path = pathNumber + 1;
65                  pathAdded = true;
66              }
67          }
68      } while (pathAdded);
69
70
71      int orginalPlayerPathNumber = tileSet.get(playerTileNum).path;
72      ArrayList<PVector> posPath = new ArrayList<>();
73      posPath.add(null);
74      for (int i = orginalPlayerPathNumber; i > 0; --i) {
75
76
77              if (playerTileNum + 1 < tileSet.size() && checkNaboPath(tileSet,
    playerTileNum, +1)) {
78                  Tile tile = getNaboTile(tileSet, playerTileNum, +1);
79                  PVector pos = new PVector(tile.xPos, tile.yPos);
80                  posPath.add(pos);
81                  playerTileNum += 1;
82              }
83              if (playerTileNum + 35 < tileSet.size() && checkNaboPath(tileSet,
    playerTileNum, +35)) {
84                  Tile tile = getNaboTile(tileSet, playerTileNum, +35);
85                  PVector pos = new PVector(tile.xPos, tile.yPos);
86                  posPath.add(pos);
87                  playerTileNum += 35;
88              }
89              if (playerTileNum + 36 < tileSet.size() && checkNaboPath(tileSet,
    playerTileNum, +36)) {
90                  Tile tile = getNaboTile(tileSet, playerTileNum, +36);
```

```
 91                  PVector pos = new PVector(tile.xPos, tile.yPos);
 92                  posPath.add(pos);
 93                  playerTileNum += 36;
 94              }
 95              if (playerTileNum + 37 < tileSet.size() && checkNaboPath(tileSet,
     playerTileNum, +37)) {
 96                  Tile tile = getNaboTile(tileSet, playerTileNum, +37);
 97                  PVector pos = new PVector(tile.xPos, tile.yPos);
 98                  posPath.add(pos);
 99                  playerTileNum += 37;
100              }
101              if (playerTileNum - 35 > 0 && checkNaboPath(tileSet,
     playerTileNum, -35)) {
102                  Tile tile = getNaboTile(tileSet, playerTileNum, -35);
103                  PVector pos = new PVector(tile.xPos, tile.yPos);
104                  posPath.add(pos);
105                  playerTileNum -= 35;
106              }
107              if (playerTileNum - 36 > 0 && checkNaboPath(tileSet,
     playerTileNum, -36)) {
108                  Tile tile = getNaboTile(tileSet, playerTileNum, -36);
109                  PVector pos = new PVector(tile.xPos, tile.yPos);
110                  posPath.add(pos);
111                  playerTileNum -= 36;
112              }
113              if (playerTileNum - 37 > 0 && checkNaboPath(tileSet,
     playerTileNum, -37)) {
114                  Tile tile = getNaboTile(tileSet, playerTileNum, -37);
115                  PVector pos = new PVector(tile.xPos, tile.yPos);
116                  posPath.add(pos);
117                  playerTileNum -= 37;
118              }
119              if (playerTileNum - 1 > 0 && checkNaboPath(tileSet,
     playerTileNum, -1)) {
120                  Tile tile = getNaboTile(tileSet, playerTileNum, -1);
121                  PVector pos = new PVector(tile.xPos, tile.yPos);
122                  posPath.add(pos);
123                  playerTileNum -= 1;
124              }
125          }
126
127          for (int i = 0; i < tileSet.size(); ++i) {
128              tileSet.get(i).path = 10000;
129              tileSet.get(i).selectedTile = false;
130          }
131          path = posPath;
132
```

```
133     }
134
135     //This function provides the neighbor to which you choose with the variable
        extra
136     Tile getNaboTile(ArrayList<Tile> tileSet, int choosen, int extra) {
137         Tile nabo = tileSet.get(choosen + extra);
138         return nabo;
139
140     }
141
142     //this function checks if the neighbor is a tile you can back to
143     Boolean checkNaboPath(ArrayList<Tile> tileSet, int choosen, int extra) {
144         Tile selected = tileSet.get(choosen);
145         Tile nabo = tileSet.get(choosen + extra);
146
147         if (selected.path - 1 == nabo.path && nabo.Contents.equals("WATER"
        )) {
148             return true;
149         } else {
150             return false;
151         }
152     }
153
154     //This function backs up the players according to what is in the list path
155     int movePlayer(int turnCount) {
156
157         if (path.size() > 1) {
158             if (turnCount <= 0) {
159                 path.clear();
160                 path.add(null);
161                 return 0;
162             }
163             xPos = path.get(1).x;
164             yPos = path.get(1).y;
165             path.remove(1);
166             System.out.println(turnCount--);
167             return turnCount--;
168
169         } else {
170             return turnCount;
171         }
172
173
174     }
175 }
176
```

```
 1  import processing.core.PApplet;
 2
 3  import java.util.ArrayList;
 4
 5  public class NPCBoat extends Boat {
 6
 7      //---------- CONSTRUCTOR :) ----------\\
 8      public NPCBoat(PApplet p, float x, float y, float m) {
 9          super(p, x, y, m);
10      }
11
12      //----------METHODS----------\\
13
14      //This function controls the NPC turn.
15      void Turn() {
16          ///cpu og player deler åbenbart items så hvad du gøre ved dene ene sker
    der også hos den anden
17          boolean canSeeShop = false;
18          ArrayList<ShopTile> shopTileset = new ArrayList<ShopTile>();
19
20          for (int i = 0; i < showneTileSet.size() - 1; ++i) {
21              if (showneTileSet.get(i).Contents.equals("SHOP")) {
22                  shopTileset.add((ShopTile) showneTileSet.get(i));
23                  canSeeShop = true;
24              }
25          }
26          if (canSeeShop && Math.random() < 0.5) {
27              System.out.println("Valgte at shoppe");
28              int random = (int) p.random(0, shopTileset.size());
29              shop(shopTileset.get(random));
30          } else {
31              System.out.println("Valgte at rykke");
32              Move(0);
33          }
34
35
36      }
37
38      //This feature allows opponents to choose to buy or sell items
39      void shop(ShopTile shopTile) {
40
41          //finds the shop menu
42          ShopMenu shop = shopTile.shopMenu;
43          //the size of the shop items
44          int s = shop.StockInventory.size();
45
46          //booleans that tell you if you have bought or sold something
```

```
47          boolean boughtSomthing = false;
48          boolean soultSomthing = false;
49
50          //chooses with 50% whether to buy something
51          if (Math.random() < 0.5) {
52              Item item = shop.StockInventory.get((int) p.random(0, s));
53              int amount = (int) (money/item.value);
54              amount = (int) p.random(amount/2,amount);
55              buy(item, amount);
56              boughtSomthing = true;
57          }
58
59          //chooses with 50% whether to sell something
60          if (Math.random() < 0.5) {
61              Item item = shop.StockInventory.get((int) p.random(0, s));
62              int amount = item.ammount;
63              amount = (int) p.random(amount/2,amount);
64              sell(item, amount);
65              soultSomthing = true;
66          }
67
68          //show nothing is bought or sold try again
69          if (!boughtSomthing || !soultSomthing) {
70              shop(shopTile);
71          }
72      }
73
74      //This function finds its selected item and sells x amount of it
75      void sell(Item item, int amount) {
76          if (money - item.value >= 0 && item.ammount - amount >= 0) {
77              for (int i = 0; i < inventory.size(); ++i) {
78                  System.out.println("itemName: " + item.Name + "|iName: " +
    inventory.get(i).Name + "|");
79                  if (inventory.get(i).Name.equals(item.Name)) {
80
81                      inventory.get(i).ammount -= amount;
82                      item.ammount += amount;
83                      money = money + item.value;
84                      break;
85
86                  }
87              }
88
89          } else if (amount > 1) {
90              System.out.println("amount: " + amount);
91              sell(item, amount - 1);
92          } else if (amount < 1) {
```

```
 93            System.out.println("amount: " + amount);
 94            buy(item, amount - 1);
 95
 96        }
 97    }
 98
 99    //This function finds its selected item and buys x amount of it
100    void buy(Item item, int amount) {
101        if (money - item.value >= 0 && item.ammount - amount >= 0) {
102            for (int i = 0; i < inventory.size(); ++i) {
103                System.out.println("itemName: " + item.Name + "|iName: " +
    inventory.get(i).Name + "|");
104                if (inventory.get(i).Name.equals(item.Name)) {
105
106
107                    inventory.get(i).ammount += amount;
108                    item.ammount -= amount;
109                    money = money - item.value;
110                    break;
111
112                }
113            }
114        } else if (amount > 1) {
115            System.out.println("amount: " + amount);
116            buy(item, amount - 1);
117        }
118    }
119
120    //This function moves the opponents to a random place
121    void Move(int trys) {
122        int xMove = (int) p.random(-3, 3);
123        int yMove = (int) p.random(-3, 3);
124        float newX = (int) (xPos + xMove);
125        float newY = (int) (yPos + yMove);
126        boolean haveFoundMove = false;
127        if (Math.random() > 0.5) {
128
129            for (int i = 0; i < showneTileSet.size() - 1; ++i) {
130                if (showneTileSet.get(i).xPos == newX && showneTileSet.get(i).
    yPos == newY && showneTileSet.get(i).Contents.equals("WATER")) {
131                    haveFoundMove = true;
132                    xPos = newX;
133                    yPos = newY;
134                }
135            }
136
137        } else if (Math.random() > 0.5) {
```

```
138             for (int i = 0; i < showneTileSet.size() - 1; ++i) {
139                 if (showneTileSet.get(i).xPos == newX && showneTileSet.get(i).
        yPos == yPos && showneTileSet.get(i).Contents.equals("WATER")) {
140                     haveFoundMove = true;
141                     xPos = newX;
142 
143                 }
144             }
145         } else {
146             for (int i = 0; i < showneTileSet.size() - 1; ++i) {
147                 if (showneTileSet.get(i).xPos == xPos && showneTileSet.get(i).
        yPos == newY && showneTileSet.get(i).Contents.equals("WATER")) {
148                     haveFoundMove = true;
149                     yPos = newY;
150                 }
151             }
152         }
153 
154         if (haveFoundMove == false&& trys<10) {
155             Move(trys+1);
156         }
157 
158     }
159 }
160 
```

```
1  import processing.core.PApplet;
2  import processing.core.PImage;
3
4  public class MainMenu {
5      //main menu variables
6      PApplet p;
7      //quick access to all the menus and the like
8      GameBoard gb;
9      PauseMenu pauseMenu;
10     ChooseGameMenu chooseGameMenu;
11     SettingMenu settingMenu;
12     //boolean which determines whether the main menu can be viewed
13     Boolean visible = true;
14     //This variable is the one that determines whether the howTopla image
       should be viewed
15     Boolean howToPlay = false;
16     //This variable helps to scale all ui elements with the screen size
17     float scaleSize = 1;
18     //the image of the background and description of how to play
19     PImage bg, howTo;
20     //all the buttons
21     NomalButton btnPlay, btnSettings, btnHowToPlay, btnLoadGame,
       btnCloseGame, backToMain;
22
23     //---------- CONSTRUCTOR :) ----------\\
24     MainMenu(PApplet p) {
25         this.p = p;
26
27         btnPlay = new NomalButton(p, 640 - 160, 260, 150, 50, "Spil");
28         btnLoadGame = new NomalButton(p, 640 + 170 - 160, 260, 150, 50, "
       Indlæs Spil");
29         btnSettings = new NomalButton(p, 640 - 160, 340, 320, 50, "
       Indstillinger");
30         btnHowToPlay = new NomalButton(p, 640 - 160, 420, 320, 50, "
       Hvordan man spiller");
31         btnCloseGame = new NomalButton(p, 640 - 160, 500, 320, 50, "luk spil"
       );
32         chooseGameMenu = new ChooseGameMenu(p, gb, this);
33         backToMain = new NomalButton(p, 540, 600, 200, 50, "Back to Menu"
       );
34         bg = p.loadImage("startside.png");
35         howTo = p.loadImage("how.png");
36     }
37     //----------METHODS----------\\
38
39     //This function draws the menu
40     void drawMenu() {
```

```
41        if (chooseGameMenu.visible && !howToPlay) {
42            chooseGameMenu.drawMenu();
43        }
44
45        if (!chooseGameMenu.visible && !howToPlay) {
46            p.clear();
47            p.image(bg, 0, 0, p.width, p.height);
48            btnPlay.drawButton();
49            btnSettings.drawButton();
50            btnLoadGame.drawButton();
51            btnHowToPlay.drawButton();
52            btnCloseGame.drawButton();
53
54            if (btnPlay.clicked) {
55
56
57                chooseGameMenu.visible = true;
58
59                btnPlay.registrerRelease();
60            }
61
62            if (btnSettings.clicked) {
63                settingMenu.backToMainMenu = true;
64                settingMenu.visible = true;
65                visible = false;
66                btnSettings.registrerRelease();
67            }
68
69            if (btnLoadGame.clicked) {
70
71                p.selectInput("Select a file to process:", "loadedMap");
72                btnLoadGame.registrerRelease();
73            }
74
75            if (btnHowToPlay.clicked) {
76                howToPlay = true;
77                btnHowToPlay.registrerRelease();
78            }
79
80            if (btnCloseGame.clicked) {
81                p.exit();
82                btnCloseGame.registrerRelease();
83
84            }
85
86        } else if (howToPlay) {
87            p.image(howTo, 0, 0, p.width, p.height);
```

```
88            backToMain.drawButton();
89            if (backToMain.clicked) {
90               howToPlay = false;
91               backToMain.registrerRelease();
92            }
93         }
94
95
96      }
97
98      //This function for all buttons and look to fit with the screen size
99      void reSizeMainMenu() {
100
101        settingMenu.reSizeBtn(scaleSize, btnPlay);
102        settingMenu.reSizeBtn(scaleSize, btnSettings);
103        settingMenu.reSizeBtn(scaleSize, btnLoadGame);
104        settingMenu.reSizeBtn(scaleSize, btnHowToPlay);
105        settingMenu.reSizeBtn(scaleSize, btnCloseGame);
106
107        settingMenu.reSizeBtn(scaleSize, chooseGameMenu.btnBackToMenu);
108        settingMenu.reSizeBtn(scaleSize, chooseGameMenu.btnLoadGame);
109        settingMenu.reSizeBtn(scaleSize, chooseGameMenu.btnNewGame);
110        settingMenu.reSizeBtn(scaleSize, chooseGameMenu.btnPlay);
111        settingMenu.reSizeBtn(scaleSize, backToMain);
112     }
113
114     //This function allows you to click on the menu
115     void menuMouseClick(float mx, float my) {
116        if (!chooseGameMenu.visible && !howToPlay) {
117
118           btnPlay.registerClick(mx, my);
119
120           btnSettings.registerClick(mx, my);
121           btnLoadGame.registerClick(mx, my);
122           btnHowToPlay.registerClick(mx, my);
123           btnCloseGame.registerClick(mx, my);
124        } else if (!howToPlay) {
125           if(!chooseGameMenu.needNewGame)
126           chooseGameMenu.btnPlay.registerClick(mx, my);
127           chooseGameMenu.btnLoadGame.registerClick(mx, my);
128           chooseGameMenu.btnBackToMenu.registerClick(mx, my);
129           chooseGameMenu.btnNewGame.registerClick(mx, my);
130        } else if (howToPlay) {
131           backToMain.registerClick(mx, my);
132        }
133
134     }
```

```
135   }
136
```

```
1  import processing.core.PApplet;
2
3  import java.util.ArrayList;
4
5  public class ShopMenu {
6      //the variables
7      PApplet p;
8      //menu variables
9      Boolean visible = false;
10     //list of standard inventory
11     ArrayList<Item> StockInventory = new ArrayList<Item>();
12     //all the buttons
13     NomalButton btnBuyBa, btnBuyRum, btnBuyEye, btnSellBa, btnSellRum,
   btnSellEye, btnCloseShop;
14     //the shop's prices
15     float bananPrices, eyepachPrices, rumPrices;
16
17     //---------- CONSTRUCTOR :) ----------\\
18     ShopMenu(PApplet p) {
19         this.p = p;
20         int xpos = 830, ypos = 150;
21         Item Banana = new Item(10, (int) p.random(0, 100), "Banana");
22         Item Rum = new Item(50, (int) p.random(0, 100), "Rum");
23         Item Eyepatch = new Item(30, (int) p.random(0, 100), "Eyepatch");
24
25         StockInventory.add(Banana);
26         StockInventory.add(Rum);
27         StockInventory.add(Eyepatch);
28
29         btnBuyEye = new NomalButton(p, xpos + 20, 270, 160, 50, "Køb en
   klap");
30         btnBuyRum = new NomalButton(p, xpos + 20, 360, 160, 50, "Køb en
   rom");
31         btnBuyBa = new NomalButton(p, xpos + 20, 460, 160, 50, "Køb en
   Banan");
32
33         btnSellEye = new NomalButton(p, xpos + 400 - 180, 270, 160, 50, "sælg
   en klap");
34         btnSellRum = new NomalButton(p, xpos + 400 - 180, 360, 160, 50, "slæg
    en rom");
35         btnSellBa = new NomalButton(p, xpos + 400 - 180, 460, 160, 50, "sælg
   en Banan");
36
37         btnCloseShop = new NomalButton(p, xpos + 20, 580, 360, 50, "luk
   butik");
38         bananPrices = determinePrice(StockInventory.get(0));
39         rumPrices = determinePrice(StockInventory.get(1));
```

```
40          eyepachPrices = determinePrice(StockInventory.get(2));
41      }
42      //----------METHODS----------\\
43
44      //this function selects the price of the shop's items
45      float determinePrice(Item item) {
46          float itemPrices = item.value;
47          float extra = p.random(-(itemPrices / 2), (itemPrices / 2));
48          item.value = itemPrices + extra;
49          return itemPrices + extra;
50
51      }
52
53      //This function draws the menu
54      void drawMenu(Player player, float s) {
55          float xpos = 830 * s, ypos = 150 * s;
56          p.fill(209, 166, 96, 255);
57          p.rect(s * 830, s * 150, s * 400, s * 500);
58          p.fill(0);
59          p.textSize(32 * s);
60          p.text("Shop", xpos + 200 * s - p.textWidth("Shop") / 2, ypos + 50 * s);
61          p.textSize(16 * s);
62          ArrayList<Item> t = StockInventory;
63
64          //buy
65          p.text(t.get(2).Name + ": " + t.get(2).ammount + " Køb/sælg for: " +
    eyepachPrices, xpos + 20 * s, ypos + 100 * s);
66          btnBuyEye.drawButton();
67          if (btnBuyEye.clicked) {
68              buyItem(StockInventory.get(2), player, btnBuyEye, eyepachPrices);
69              btnBuyEye.registrerRelease();
70          }
71
72          //sell
73          btnSellEye.drawButton();
74          if (btnSellEye.clicked) {
75              SellItem(player.inventory.get(2), player, btnSellEye, 1, eyepachPrices);
76              btnSellEye.registrerRelease();
77          }
78
79
80          p.text(t.get(1).Name + ": " + t.get(1).ammount + " Køb/sælg for: " +
    rumPrices, xpos + 20 * s, ypos + 200 * s);
81          btnBuyRum.drawButton();
82          if (btnBuyRum.clicked) {
83              buyItem(StockInventory.get(1), player, btnBuyRum, rumPrices);
84              btnBuyRum.registrerRelease();
```

```
 85          }
 86
 87          btnSellRum.drawButton();
 88          if (btnSellRum.clicked) {
 89              SellItem(player.inventory.get(1), player, btnSellRum, 1, rumPrices);
 90              btnSellRum.registrerRelease();
 91          }
 92
 93          p.text(t.get(0).Name + ": " + t.get(0).ammount + " Køb/sælg for: " +
     bananPrices, xpos + 20 * s, ypos + 300 * s);
 94          btnBuyBa.drawButton();
 95          if (btnBuyBa.clicked) {
 96              buyItem(StockInventory.get(0), player, btnBuyBa, bananPrices);
 97              btnBuyBa.registrerRelease();
 98          }
 99
100          btnSellBa.drawButton();
101          if (btnSellBa.clicked) {
102              SellItem(player.inventory.get(0), player, btnSellBa, 1, bananPrices);
103              btnSellBa.registrerRelease();
104          }
105
106          btnCloseShop.drawButton();
107          if (btnCloseShop.clicked) {
108
109              visible = false;
110              btnCloseShop.registrerRelease();
111          }
112
113
114      }
115
116      //This function for all buttons and look to fit with the screen size
117      void reSizeShopMenu(float s) {
118          btnBuyBa.size = s;
119          btnBuyRum.size = s;
120          btnBuyEye.size = s;
121          btnSellBa.size = s;
122          btnSellRum.size = s;
123          btnSellEye.size = s;
124          btnCloseShop.size = s;
125
126      }
127
128      //This function allows you to click on the menu
129      void shopMenuClicked() {
130          btnBuyBa.registerClick(p.mouseX, p.mouseY);
```

```
131         btnBuyEye.registerClick(p.mouseX, p.mouseY);
132         btnBuyRum.registerClick(p.mouseX, p.mouseY);
133
134         btnSellBa.registerClick(p.mouseX, p.mouseY);
135         btnSellEye.registerClick(p.mouseX, p.mouseY);
136         btnSellRum.registerClick(p.mouseX, p.mouseY);
137
138         btnCloseShop.registerClick(p.mouseX, p.mouseY);
139     }
140
141     //This function allows you to purchase items
142     void buyItem(Item item, Player player, NomalButton btn, float itemPrices
        ) {
143         if (player.money - itemPrices >= 0 && item.ammount - 1 >= 0 && btn.
        clicked) {
144             for (int i = 0; i < player.inventory.size(); ++i) {
145                 System.out.println("itemName: " + item.Name + "|iName: " +
        player.inventory.get(i).Name + "|");
146                 if (player.inventory.get(i).Name.equals(item.Name)) {
147
148
149                     ++player.inventory.get(i).ammount;
150                     --item.ammount;
151                     player.money = player.money - itemPrices;
152                     btn.registrerRelease();
153                     break;
154
155                 }
156             }
157         }
158     }
159
160     //This function allows you to sell items
161     void SellItem(Item item, Player player, NomalButton btn, int amount, float
         itemPrices) {
162
163         int sellAmount = item.ammount - amount;
164         if (sellAmount >= 0) {
165             for (int i = 0; i < StockInventory.size(); ++i) {
166                 System.out.println("itemName: " + item.Name + "|iName: " +
        player.inventory.get(i).Name + "|");
167                 if (StockInventory.get(i).Name.equals(item.Name)) {
168
169                     item.ammount -= amount;
170                     StockInventory.get(i).ammount += amount;
171                     player.money = player.money + itemPrices;
172                     btn.registrerRelease();
```

```
173                     break;
174
175                 }
176             }
177
178         }
179     }
180
181
182 }
183
184
185
186
```

```java
import processing.core.PApplet;

public class ShopTile extends Tile {
    //shop menu the tile
    ShopMenu shopMenu;

    //---------- CONSTRUCTOR :) ----------\\
    public ShopTile(PApplet p, String C, float x, float y) {
        super(p, C, x, y);
        Contents = "SHOP";
        shopMenu = new ShopMenu(p);
    }
    //----------METHODS----------\\

    //This draws the shop menu
    public void drawShopMenu(Player player, float s) {
        if (shopMenu.visible) {
            shopMenu.drawMenu(player, s);
        }
    }

    //resider the shop and make the shop menu visble/hide it away
    public void showShop(float s) {
        shopMenu.reSizeShopMenu(s);
        shopMenu.visible = !shopMenu.visible;
    }

    //this function do so you can click on the shop
    public void clickShop() {
        shopMenu.shopMenuClicked();
    }

    //this function change the image on the tile
    @Override
    public void setTileImage() {
        tileImage = p.loadImage("shop.png");
    }
}

```

```
 1  import processing.core.PApplet;
 2  import processing.core.PImage;
 3  import processing.core.PVector;
 4
 5  import java.util.ArrayList;
 6
 7  public class GameBoard {
 8      //all the variables on the playing board
 9      PApplet p;
10      //this boolean determines whether to use alcohol on the board or not
11      boolean showTileImage = true;
12      //these are all tiles on the map
13      ArrayList<Tile> tileSet = new ArrayList<Tile>();
14      //all the buttons
15      NomalButton btnMenu, btnMap, btnAcceptRul, btnBackToMenu;
16      //pause menu
17      public PauseMenu pauseMenu;
18      //quick access to settings
19      SettingMenu settingMenu;
20      //this class is responsible for creating the folder and saving the game
21      SaveManger saveManger;
22      //the players
23      public Player player;
24
25      //this boat will be the winners eventually
26      Boat theWinner;
27      //this variable tells if the players have won
28      boolean playerWon;
29      //This boolean tells you whether the folder should be drawn or the tileset
30      boolean drawmap = false;
31      //Devconsolen
32      DevConsole devConsole;
33      //the list of all opponents
34      ArrayList<NPCBoat> NPCBoatArrayList = new ArrayList<NPCBoat>();
35      //max rounds
36      int maxRounds = 300;
37      //number of opponents
38      int numbersOfCpus = 3;
39      //round you are reached
40      int roundCount = 0;
41      //the players turns
42      int turnCount;
43      //This variable helps to scale all ui elements with the screen size
44      float scaleSize = 1;
45      //This boolean determines if rolling is in progress
46      Boolean rul = true;
47      //Whether the game board is visible
```

```
48      Boolean visible = false;
49      //This boolean tells if the trip is over
50      Boolean turnEnded = false;
51      //This list has good cpu positions for the start
52      ArrayList<PVector> cpuPos = new ArrayList<>();
53      //the background to the game board
54      PImage bg;
55      //This variable determines the roll animation
56      int rulleAnimation = 0;
57
58      //---------- CONSTRUCTOR :) ----------\\
59      GameBoard(PApplet p, PauseMenu pauseMenu) {
60
61          this.p = p;
62          saveManger = new SaveManger(p);
63          saveManger.gb = this;
64          saveManger.generateGame(32, tileSet);
65
66
67          this.pauseMenu = pauseMenu;
68          startGame(numbersOfCpus, cpuPos);
69          btnMenu = new NomalButton(p, p.width - 60, 0, 60, 60, "-\n-\n-");
70          btnMap = new NomalButton(p, 270, 650, 450, 50, "map");
71          btnAcceptRul = new NomalButton(p, 270, 650, 450, 50, "Rul");
72          btnBackToMenu = new NomalButton(p, 270, 650, 450, 50, "Tilbage til
    menu");
73          pauseMenu.gb = this;
74          pauseMenu.mainMenu.gb = this;
75          pauseMenu.mainMenu.chooseGameMenu.gb = this;
76
77          settingMenu = pauseMenu.settingMenu;
78          settingMenu.gb = this;
79          settingMenu.tfNumbersOfPlayers.input = "" + numbersOfCpus;
80          settingMenu.tfMaxRound.input = "" + maxRounds;
81          devConsole = new DevConsole(p, this);
82          settingMenu.sm = saveManger;
83          settingMenu.tfGenNum.input = "" + saveManger.increment * 100;
84
85      }
86      //----------METHODS----------\\
87
88      //This function starts the game by making all the boats
89      void startGame(int antalCPU, ArrayList<PVector> cpuPos) {
90
91          int playerPos = (int) p.random(0, cpuPos.size());
92          player = new Player(p, cpuPos.get(playerPos).x, cpuPos.get(playerPos).y
    , 500);
```

```
 93          cpuPos.remove(playerPos);
 94          player.generateInventory();
 95          player.boatPic = p.loadImage("Skibet32.png");
 96          //turnList.add(new Turn(p,player));
 97          // turnCount = (int) p.random(0, 7);
 98          turnCount = 1;
 99          for (int i = 0; i < antalCPU; ++i) {
100              int cpurPos = (int) p.random(0, cpuPos.size());
101              int x = (int) cpuPos.get(cpurPos).x, y = (int) cpuPos.get(cpurPos).y;
102              cpuPos.remove(cpurPos);
103              System.out.println("x: " + x + " y: " + y);
104              NPCBoatArrayList.add(new NPCBoat(p, x, y, 500));
105
106          }
107
108          for (int i = 0; i < NPCBoatArrayList.size(); ++i) {
109
110              NPCBoatArrayList.get(i).generateInventory();
111              NPCBoatArrayList.get(i).boatPic = p.loadImage("piratbad.png");
112              // turnList.add(new Turn(p,cpuArrayList.get(i)));
113          }
114
115
116      }
117
118      //This function the drawing board
119      void drawBoard() {
120
121          if (visible && !devConsole.visible && maxRounds > roundCount) {
122
123              p.clear();
124              //show background image is not there so it loads it
125              if (bg != null) {
126                  p.image(bg, 0, 0, p.width, p.height);
127              } else {
128                  p.background(200);
129                  bg = p.loadImage("bgofgame.png");
130              }
131
132              //Drawing the ui
133              p.textSize(16 * scaleSize);
134              p.fill(200);
135              p.rect(0, 0, p.width, 60 * scaleSize);
136              String info = "Round: " + roundCount + "/" + maxRounds + "\n
    Turn: " + turnCount;
137              p.fill(0);
138              p.text(info, (p.width / 2 - p.textWidth(info) / 2), 30 * scaleSize);
```

```
139         ArrayList<Item> t = player.inventory;
140         p.fill(200);
141
142         p.fill(0);
143         p.text("\nTaske: " + "\nPenge" + player.money
144                 + "\n" + t.get(0).Name + t.get(0).ammount
145                 + "\n" + t.get(1).Name + t.get(1).ammount
146                 + "\n" + t.get(2).Name + t.get(2).ammount
147                 + "\n ----------\n"
148
149             , 880 * scaleSize, 150 * scaleSize);
150
151         //fills the players list of tiles they can see
152         player.fillUpShownTiles(tileSet);
153         //fills the list up of tiles the npcs can see
154         for (int j = 0; j < NPCBoatArrayList.size(); ++j) {
155             NPCBoatArrayList.get(j).fillUpShownTiles(tileSet);
156             if (rulleAnimation > 0) {
157                 Rul();
158                 rulleAnimation--;
159             }
160         }
161
162         //draws the buttons
163         btnMenu.drawButton();
164         btnMap.drawButton();
165
166         //perform the functions of the buttons when clicked
167         if (btnMenu.clicked) {
168             aktiverPauseMenu();
169         }
170         if (btnMap.clicked) {
171             drawmap = !drawmap;
172             btnMap.registrerRelease();
173         }
174         //draws the tiles that players can see
175         ArrayList<Tile> showneTileSet = player.showneTileSet;
176         if (!drawmap) {
177             for (int i = 0; i < showneTileSet.size(); ++i) {
178                 p.pushMatrix();
179                 float posX = showneTileSet.get(i).xPos;
180                 float posY = showneTileSet.get(i).yPos;
181
182                 if (i < 5) {
183
184                     //225*scaleSize
185                     posX = 2.5f;
```

```
186                     posY = i + 1.5f;
187                 } else if (i < 10) {
188                     posX = 3.5f;
189                     posY = i - 3.5f;
190                 } else if (i < 15) {
191                     posX = 4.5f;
192                     posY = i - 8.5f;
193                 } else if (i < 20) {
194                     posX = 5.5f;
195                     posY = i - 13.5f;
196                 } else {
197                     posX = 6.5f;
198                     posY = i - 18.5f;
199                 }
200                 //show tilen does not have a picture it adds it here
201                 if (showTileImage && showneTileSet.get(i).tileImage == null) {
202                     showneTileSet.get(i).setTileImage();
203                 }
204
205                 showneTileSet.get(i).Display(posX, posY, (int) (85 * scaleSize),
    showTileImage, scaleSize);
206                 showneTileSet.get(i).drawShopMenu(player, scaleSize);
207                 showneTileSet.get(i).checkIfMouseOver(posX, posY, (int) (85 *
    scaleSize));
208
209                 //drawing the npc boats
210                 drawShips(player, i, posX, posY);
211                 for (int j = 0; j < NPCBoatArrayList.size(); ++j) {
212                     drawShips(NPCBoatArrayList.get(j), i, posX, posY);
213                 }
214                 p.popMatrix();
215
216                 //show the shop is clicked then open the menu
217                 if (showneTileSet.get(i).cliked) {
218                     if (showneTileSet.get(i).Contents.equals("SHOP")) {
219                         //klikker på shop
220                         player.findPath(showneTileSet.get(i), tileSet);
221                         System.out.println("ShopTime");
222                         showneTileSet.get(i).showShop(scaleSize);
223
224
225                     } else if (!showneTileSet.get(i).Contents.equals("BORDER"
    )) {
226                         showneTileSet.get(i).selectedTile = true;
227                         if (showneTileSet.get(i).selectedTile) {
228                             player.findPath(showneTileSet.get(i), tileSet);
229                         }
```

```
230                     }
231                     showneTileSet.get(i).cliked = false;
232                 }
233                 turnCount = player.movePlayer(turnCount);
234
235                 if (turnCount <= 0 && !rul) {
236                     turnEnded = true;
237
238                 }
239
240
241             }
242         } else {
243             //draws the map of the map
244             for (int i = 0; i < tileSet.size(); ++i) {
245                 p.pushMatrix();
246                 p.translate(225 * scaleSize, 140 * scaleSize);
247                 Tile tile = tileSet.get(i);
248                 tile.Display(tile.xPos, tile.yPos, (int) (12 * scaleSize), false,
    scaleSize);
249                 p.fill(201, 0, 0);
250                 if (tile.xPos == player.xPos && tile.yPos == player.yPos) {
251
252                     float s = (int) (12 * scaleSize);
253                     p.rect(s * tile.xPos, s * tile.yPos, s, s);
254                 }
255                 p.popMatrix();
256             }
257
258         }
259         //when the players' turn is over it goes through all the npcs turn and
    allows the players to roll the dice
260         if (turnEnded) {
261             p.textSize(60 * scaleSize);
262             p.fill(200, 200, 200, 200);
263             p.rect(0, p.height / 2 - 60 * scaleSize, p.width, 80 * scaleSize);
264             p.fill(0);
265             p.text("VENTER PÅ MODSPILLERNES TUR", p.width / 2 - p.
    textWidth("VENTER " +
266                     "PÅ MODSPILLERNES TUR") / 2, p.height / 2);
267
268             for (int j = 0; j < NPCBoatArrayList.size(); ++j) {
269                 NPCBoatArrayList.get(j).Turn();
270                 System.out.println("cpu number: " + j);
271             }
272             if (!rul)
273                 ++roundCount;
```

```
274             turnEnded = false;
275             rul = true;
276         }
277         Rul();
278
279     //draws the dev console if it is visible
280     } else if (devConsole.visible) {
281         devConsole.display(scaleSize);
282     //checks if the game is over and shows it checks who won
283     } else if (maxRounds <= roundCount) {
284         chooseWinner();
285         p.textSize(14 * scaleSize);
286         if (playerWon) {
287             String s = "Du vandt!";
288             p.text(s, p.width / 2 - p.textWidth(s) / 2, p.height / 2);
289         } else {
290             String s = "Du Tabte!";
291             p.text(s, p.width / 2 - p.textWidth(s) / 2, p.height / 2);
292         }
293         btnBackToMenu.drawButton();
294
295         if (btnBackToMenu.clicked) {
296             visible = false;
297
298             main.mainMenu.visible = true;
299             main.mainMenu.chooseGameMenu.needNewGame = true;
300             saveManger.increment += p.random(-0.02f, 0.02f);
301
302             main.mainMenu.chooseGameMenu.needNewGame = true;
303             btnBackToMenu.registrerRelease();
304         }
305     }
306 }
307
308 //This function for all buttons and look to fit with the screen size
309 void reSizeGamebord() {
310     settingMenu.reSizeBtn(scaleSize, btnMenu);
311     settingMenu.reSizeBtn(scaleSize, btnMap);
312     settingMenu.reSizeBtn(scaleSize, btnAcceptRul);
313     settingMenu.reSizeBtn(scaleSize, btnBackToMenu);
314     settingMenu.reSizeFT(scaleSize, devConsole.textField);
315
316 }
317
318 //This function draws a boat
319 void drawShips(Boat boat, int i, float posX, float posY) {
320     ArrayList<Tile> showneTileSet = player.showneTileSet;
```

```
321         if (boat.xPos == showneTileSet.get(i).xPos && boat.yPos ==
        showneTileSet.get(i).yPos) {
322
323             boat.displayBoat(posX, posY, (int) (85 * scaleSize));
324         }
325     }
326
327     //This function activates the pause menu
328     void aktiverPauseMenu() {
329         pauseMenu.visible = true;
330         visible = false;
331         btnMenu.registrerRelease();
332     }
333
334     //This function activates the dev console
335     void akitverDevconsole() {
336         devConsole.visible = !devConsole.visible;
337
338     }
339
340     //This function allows you to click on the board
341     void boardmouseClicked() {
342         ArrayList<Tile> showneTileSet = player.showneTileSet;
343         devConsole.mouseClick();
344         if (!rul) {
345             btnMenu.registerClick(p.mouseX, p.mouseY);
346             btnMap.registerClick(p.mouseX, p.mouseY);
347             if (!drawmap)
348                 if (turnEnded == false) {
349                     for (int i = 0; i < showneTileSet.size(); ++i) {
350                         float posX = showneTileSet.get(i).xPos;
351                         float posY = showneTileSet.get(i).yPos;
352
353                         if (i < 5) {
354
355                             //225*scaleSize
356                             posX = 2.5f;
357                             posY = i + 1.5f;
358                         } else if (i < 10) {
359                             posX = 3.5f;
360                             posY = i - 3.5f;
361                         } else if (i < 15) {
362                             posX = 4.5f;
363                             posY = i - 8.5f;
364                         } else if (i < 20) {
365                             posX = 5.5f;
366                             posY = i - 13.5f;
```

```
367                     } else {
368                         posX = 6.5f;
369                         posY = i - 18.5f;
370                     }/**/
371
372                     showneTileSet.get(i).clickShop();
373                     showneTileSet.get(i).checkIfCliked((int) posX, (int) posY, (
    int) (100 * scaleSize));
374
375                 }
376             }
377         } else {
378             btnAcceptRul.registerClick(p.mouseX, p.mouseY);
379         }
380         if (maxRounds <= roundCount) {
381             btnBackToMenu.registerClick(p.mouseX, p.mouseY);
382         }
383     }
384
385     //This feature rolls the dice
386     void Rul() {
387         if (rul) {
388             int terningTal = (int) (p.random(1, 6));
389             System.out.println(terningTal);
390             PImage img2 = p.loadImage("terning0.png");
391             PImage img = p.loadImage("terning" + p.str(terningTal) + ".png");
392             p.fill(200, 200, 200, 200);
393             p.rect(0, 60 * scaleSize, p.width, p.height);
394             p.image(img2, p.width / 2 - 100 * scaleSize, p.height / 2 - 100 *
    scaleSize, 200 * scaleSize, 200 * scaleSize);
395             p.image(img, p.width / 2 - 100 * scaleSize, p.height / 2 - 100 *
    scaleSize, 200 * scaleSize, 200 * scaleSize);
396             btnAcceptRul.drawButton();
397             if (btnAcceptRul.clicked) {
398                 rul = false;
399                 turnCount = terningTal;
400                 btnAcceptRul.registrerRelease();
401             }
402         }
403
404
405     }
406
407     //This feature allows the dev console text field to detect what you are
    clicking on
408     void keyTyped() {
409         if (visible) {
```

```
410             devConsole.keybordTyped();
411         }
412     }
413
414     //Your feature finds the winners of the game
415     void chooseWinner() {
416         Boat winner = NPCBoatArrayList.get(0);
417         for (int i = 0; i < NPCBoatArrayList.size(); ++i) {
418             Boat boat = NPCBoatArrayList.get(i);
419             if (winner != boat) {
420                 if (winner.calRating() < boat.calRating()) {
421                     winner = boat;
422                     i = 0;
423                 }
424             }
425         }
426
427         System.out.println("cpu " + winner.calRating() + " player" + player.
    calRating());
428
429         if (winner.calRating()  <= player.calRating()) {
430             theWinner = player;
431             playerWon = true;
432         } else {
433             theWinner = winner;
434             playerWon = false;
435         }
436     }
437
438 }
439
440
```

```
 1  import processing.core.PApplet;
 2
 3  public class PauseMenu {
 4
 5      //menu variables
 6      PApplet p;
 7      //Boolean that tells about the menu can be seen
 8      Boolean visible = false;
 9      //quick access to the other menus
10      GameBoard gb;
11      MainMenu mainMenu;
12      SettingMenu settingMenu;
13      //all the buttons
14      NomalButton btnResume, btnSettings, btnExitToMain, btnSave;
15
16      //---------- CONSTRUCTOR :) ----------\\
17      PauseMenu(PApplet p, SettingMenu settingMenu, MainMenu mainMenu) {
18          this.p = p;
19          this.mainMenu = mainMenu;
20          this.settingMenu = settingMenu;
21
22
23          mainMenu.settingMenu = settingMenu;
24          mainMenu.pauseMenu = this;
25          settingMenu.pauseMenu = this;
26
27          btnResume = new NomalButton(p, 640 - 160, 100, 320, 50, "Forsæt");
28          btnSettings = new NomalButton(p, 640 - 160, 160, 320, 50, "
    Indstillinger");
29          btnSave = new NomalButton(p, 640 - 160, 220, 320, 50, "Gem Spil");
30          btnExitToMain = new NomalButton(p, 640 - 160, 280, 320, 50, "Tilbage
     Til Menu");
31
32
33      }
34      //----------METHODS----------\\
35
36      //This function draws the menu
37      void drawMenu() {
38          p.clear();
39          p.image(main.bg,0,0,p.width,p.height);
40          btnResume.drawButton();
41          btnSettings.drawButton();
42          btnExitToMain.drawButton();
43          btnSave.drawButton();
44
45          if (btnResume.clicked) {
```

```
46          aktiverGameBord();
47      }
48
49      if (btnSettings.clicked) {
50          settingMenu.backToMainMenu = false;
51          settingMenu.visible = true;
52          visible = false;
53          btnSettings.registrerRelease();
54      }
55
56      if (btnExitToMain.clicked) {
57          btnExitToMain.registrerRelease();
58          mainMenu.chooseGameMenu.needNewGame = false;
59          mainMenu.visible = true;
60          visible = false;
61      }
62
63      if (btnSave.clicked) {
64          gb.saveManger.saveGame(32, gb.tileSet);
65          btnSave.registrerRelease();
66      }
67
68
69  }
70
71  //This function for all buttons and look to fit with the screen size
72  void reSizePauseMenu() {
73
74      float size = settingMenu.displayResolution[settingMenu.
    displayResolutionInt].z;
75      settingMenu.reSizeBtn(size, btnResume);
76      settingMenu.reSizeBtn(size, btnSettings);
77      settingMenu.reSizeBtn(size, btnExitToMain);
78      settingMenu.reSizeBtn(size, btnSave);
79  }
80
81  //This function allows you to click on the menu
82  void clickMenu(float mx, float my) {
83      btnResume.registerClick(mx, my);
84      btnSettings.registerClick(mx, my);
85      btnExitToMain.registerClick(mx, my);
86      btnSave.registerClick(mx, my);
87  }
88
89  //This function activates the game board
90  void aktiverGameBord() {
91      gb.visible = true;
```

```
92          gb.reSizeGamebord();
93          visible = false;
94          btnResume.registrerRelease();
95      }
96  }
97
```

```
 1  import processing.core.PApplet;
 2
 3  public class TextField {
 4
 5      //button "positions" and "size".
 6      float positionX, positionY, sizeX, sizeY;
 7      //button size and position on the screen
 8      float realPositionX, realPositionY, realSizeX, realSizeY;
 9      //mouse x and y postion
10      float mouseX, mouseY;
11      //The size. This size is multiplied by "positions" and "size"
12      // to get the positions and size of the screen.
13      // that way the game can font window size without destroying anything
14      float size = 1;
15      //this string is the tilt
16      String textfieldTitle;
17      //this is the input
18      String input = "";
19      //boolean if the TextField is clicked
20      boolean clicked = false;
21      //whether it uses only spoken or not
22      boolean acceptLetters = true;
23
24      PApplet p;
25
26      //---------- CONSTRUCTOR :) ----------\\
27      TextField(PApplet papp, int posX, int posY, int sizeX, int sizeY, String text
    ) {
28          p = papp;
29          positionX = posX;
30          positionY = posY;
31          this.sizeX = sizeX;
32          this.sizeY = sizeY;
33          this.textfieldTitle = text;
34          realSizeY = sizeY;
35          realSizeX = sizeX;
36
37          realPositionY = posY;
38          realPositionX = posX;
39      }
40      //----------METHODS----------\\
41
42      //This function checks if the mouse is over the feild and is clicked change it
    clicked to true
43      void registerClick(float mouseX, float mouseY) {
44          this.mouseX = mouseX;
45          this.mouseY = mouseY;
```

```
46          if (mouseX > positionX &&
47                  mouseX < positionX + sizeX &&
48                  mouseY > positionY &&
49                  mouseY < positionY + sizeY) {
50              clicked = !clicked;
51          } else {
52              clicked = false;
53          }
54
55
56      }
57
58      //drawing the field
59      void drawField() {
60
61          positionX = realPositionX * size;
62          positionY = realPositionY * size;
63          sizeX = realSizeX * size;
64          sizeY = realSizeY * size;
65
66          p.stroke(1, 46, 74, 100);
67          if (clicked) {
68              p.fill(227, 225, 252, 250);
69          } else {
70              p.fill(200);
71          }
72
73
74          p.rect(positionX, positionY, sizeX, sizeY);
75
76          p.fill(0);
77          p.textSize(16 * size);
78          p.text(input, positionX + (sizeX / 16), positionY + (sizeY / 2));
79          p.text(textfieldTitle, positionX, positionY);
80      }
81
82      //this function lets you change the input
83      void keyinput(char key) {
84
85          if (clicked) {
86              if (key == p.BACKSPACE && input.length() > 0) {
87
88                  input = input.substring(0, input.length() - 1);
89              } else {
90                  if (acceptLetters || (key >= '0' && key <= '9'))
91                      input = input + key;
92              }
```

```
 93
 94         }
 95         PApplet.println(input);
 96     }
 97
 98
 99 }
100
```

```
 1  import processing.core.PApplet;
 2  import processing.core.PVector;
 3
 4  public class DevConsole extends PApplet {
 5      //variable consoles
 6      PApplet p;
 7
 8      //this variable tells if the console is visible
 9      boolean visible = false;
10
11      //quick access to the game board
12      GameBoard gameBoard;
13      //textfield to which you type the commands
14      TextField textField;
15      //string contains the information that has occurred under the console
16      String display = "...";
17      //the position of the console
18      PVector displayPos = new PVector(1, 1);
19      //the size of the text
20      float textSize = 30;
21
22      //---------- CONSTRUCTOR :) ----------\\
23      DevConsole(PApplet p, GameBoard gameBoard) {
24          this.p = p;
25          this.gameBoard = gameBoard;
26          textField = new TextField(p, p.width / 12 / 2, p.height - p.height / 5, p.
    width - p.width / 12, p.width / 12 / 2, "DevConsole");
27          displayPos = new PVector(1, 1);
28      }
29      //----------METHODS----------\\
30
31      //this function draws the console
32      void display(float s) {
33          textSize = 30 * s;
34          displayPos = new PVector(p.width / 12 / 2 + textSize, p.height - p.height
     / 4 - textSize );
35          p.fill(0);
36          p.rect((p.width / 12 / 2), 0, (p.width - p.width / 12), (p.height - p.height / 4
    ));
37          textField.drawField();
38          textField.size = s;
39          p.fill(255);
40
41          p.textSize(textSize);
42
43          p.text(display, displayPos.x, displayPos.y);
44          p.textSize(12 * s);
```

```
45          p.fill(0);
46      }
47
48
49      //this feature lets you click on the console
50      void mouseClick() {
51          if (visible) {
52              textField.registerClick(p.mouseX, p.mouseY);
53          }
54      }
55
56      //this feature lets you type in the console
57      void keybordTyped() {
58          if (visible) {
59
60              if (p.key == ENTER && textField.input.length() > 0) {
61                  textField.clicked = false;
62                  display =  textField.input;
63
64                  comandos();
65                  textField.input = "";
66              } else if (p.key == BACKSPACE && textField.input.length() > 0) {
67                  textField.input = textField.input.substring(0, textField.input.length
    () - 1);
68              } else {
69                  textField.keyinput(p.key);
70              }
71              textField.clicked = true;
72          } else {
73              textField.clicked = false;
74          }
75
76      }
77
78      //This console takes your input and tries to figure out what you would do
79      void comandos() {
80          String ip = textField.input;
81          String[] letters = ip.split("-");
82          for (int i = 0; i < letters.length; ++i)
83              System.out.println("i: " + i + letters[i]);
84
85          if (letters.length > 1 && letters[0].equalsIgnoreCase("g")) {
86              if (letters.length > 2 && letters[1].equalsIgnoreCase("t")) {
87                  gameBoard.turnCount = Integer.valueOf(letters[2]);
88                  display = "Game Turns (" + gameBoard.turnCount + ")";
89
90              }
```

```
 91          }
 92
 93          if (letters.length > 1 && letters[0].equalsIgnoreCase("p")) {
 94              System.out.println("player");
 95              if (letters.length > 3 && letters[1].equalsIgnoreCase("pos")) {
 96                  gameBoard.player.xPos = Integer.valueOf(letters[2]);
 97                  gameBoard.player.yPos = Integer.valueOf(letters[3]);
 98                  display = "Players postion (" + gameBoard.player.xPos + "x" +
     gameBoard.player.xPos + ")";
 99
100                  System.out.println("postion");
101              } else if (letters[1].equalsIgnoreCase("addM")) {
102                  gameBoard.player.money += Integer.valueOf(letters[2]);
103                  display += "Players money (" + gameBoard.player.money + ")";
104
105                  System.out.println("money");
106              } else if (letters[1].equalsIgnoreCase("removeM")) {
107                  gameBoard.player.money -= Integer.valueOf(letters[2]);
108                  display += "Players money (" + gameBoard.player.money + ")";
109
110              }
111
112          }
113
114
115      }
116  }
117
118
```

```
 1  import processing.core.PApplet;
 2  import processing.core.PVector;
 3  import processing.data.Table;
 4
 5  import javax.swing.*;
 6  import java.util.ArrayList;
 7
 8  public class SaveManger {
 9      //the variables
10      PApplet p;
11      //quick access to the game board
12      GameBoard gb;
13      //increment to the genoration of tiles
14      public float increment = (float) 0.09;
15      //list of possible positions for the npcs
16      ArrayList<PVector> cpuPos = new ArrayList<>();
17
18      //---------- CONSTRUCTOR :) ----------\\
19      SaveManger(PApplet p) {
20          this.p = p;
21          for (int i = 0; i < 4; ++i) {
22              int x = (int) p.random(0, 33), y = (int) p.random(0, 33);
23              PVector pos = new PVector(x, y);
24              cpuPos.add(pos);
25          }
26      }
27      //----------METHODS----------\\
28
29      //This function saves all information in the game in a csv file
30      void saveGame(int numberOfTiles, ArrayList<Tile> tileSet) {
31          System.out.println("s");
32          ArrayList<Tile> mapTiles = new ArrayList<Tile>();
33          for (int i = 0; i < tileSet.size(); ++i) {
34              String contents = tileSet.get(i).Contents;
35              if (!contents.equalsIgnoreCase("BORDER")) {
36                  mapTiles.add(tileSet.get(i));
37              }
38          }
39          Table map = new Table();
40          map.addColumn();
41          for (int i = 0; i < mapTiles.size(); ++i) {
42              int xPos = mapTiles.get(i).xPos - 1;
43              int yPos = mapTiles.get(i).yPos - 1;
44              String contens = mapTiles.get(i).Contents;
45              if (contens.equals("SAND")) {
46                  map.setString(xPos, yPos, "B");
47              } else if (contens.equals("WATER")) {
```

```
48              map.setString(xPos, yPos, "W");
49          } else if (contens.equals("SHOP")) {
50              map.setString(xPos, yPos, "S");
51          } else if (contens.equals("GRASS")) {
52              map.setString(xPos, yPos, "G");
53          }
54      }
55      map.setString(0, 32, "Round");
56      map.setString(2, 32, "Turn");
57      map.setString(4, 32, "Numbers of CPU's");
58      map.setInt(1, 32, gb.roundCount);
59      map.setInt(3, 32, gb.turnCount);
60      map.setInt(5, 32, gb.numbersOfCpus);
61      loadBoatToString(map, "Player", 33, gb.player);
62      for (int i = 0; i < gb.NPCBoatArrayList.size(); ++i) {
63          loadBoatToString(map, "CPU", 34 + i, gb.NPCBoatArrayList.get(i));
64      }
65      JFileChooser f = new JFileChooser();
66      f.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
67      f.showSaveDialog(null);
68      System.out.println(f.getCurrentDirectory());
69      try{
70          System.out.println(p.saveTable(map,f.getCurrentDirectory() + "
    GameSave"+p.day()+"d"+p.month()+"m"+p.year()+"y.csv"));
71      }catch (Exception e){
72          System.out.println(e);
73      }
74
75  }
76
77  //takes the boats and makes adds them to the csv file
78  void loadBoatToString(Table map, String titel, int column, Boat boat) {
79      map.setString(0, column, titel);
80      map.setString(1, column, String.valueOf(gb.player.money));
81      map.setInt(2, column, (int) boat.xPos);
82      map.setInt(3, column, (int) boat.yPos);
83      ArrayList<Item> inventory = boat.inventory;
84      for (int i = 0; i < inventory.size(); ++i) {
85          String name = inventory.get(i).Name;
86          float value = inventory.get(i).value;
87          int amount = inventory.get(i).ammount;
88          map.setString(4 + (i * 3), column, name);
89          map.setFloat(5 + (i * 3), column, value);
90          map.setInt(6 + (i * 3), column, amount);
91      }
92  }
93
```

```
94      //This function takes the boats from the csv file and converts it to a boat
95      void loadMapToBoat(Table map, String titel, int column, Boat boat) {
96          ArrayList<Item> inventory = boat.inventory;
97          ArrayList<Item> newInventory = new ArrayList<Item>();
98          for (int i = 0; i < inventory.size(); ++i) {
99              float value = map.getFloat(5 + (i * 3), column);
100             int amount = map.getInt(6 + (i * 3), column);
101             String name = map.getString(4 + (i * 3), column);
102             Item item = new Item(value, amount, name);
103             newInventory.add(item);
104         }
105         boat.money = map.getFloat(1, column);
106         boat.inventory = newInventory;
107
108         gb.roundCount = map.getInt(1, 32);
109         gb.turnCount = map.getInt(3, 32);
110         boat.xPos = map.getInt(2, column);
111         boat.yPos = map.getInt(3, column);
112
113
114     }
115
116     //This function takes the csv file and adds it to the game
117     void loadGame(int NumberoFTiles, Table map, ArrayList<Tile> tileSet) {
118         tileSet.clear();
119
120         for (int x = -1; x < NumberoFTiles + 3; ++x) {
121             for (int y = -1; y < NumberoFTiles + 3; ++y) {
122                 Tile t = new TerrainTile(p, "", x, y);
123                 if ((x <= 0 || y <= 0) || (33 <= x || 33 <= y)) {
124                     t.Contents = "BORDER";
125                     t.tileImage = p.loadImage("border.png");
126                 } else {
127                     int xPos = x - 1;
128                     int yPos = y - 1;
129                     String contens = map.getString(xPos, yPos);
130                     if (contens.equals("W")) {
131                         contens = "WATER";
132                     } else if (contens.equals("B")) {
133                         contens = "SAND";
134
135                     } else if (contens.equals("S")) {
136                         contens = "SHOP";
137                         t = new ShopTile(p, contens, xPos + 1, yPos + 1);
138                     } else if (contens.equals("G")) {
139                         contens = "GRASS";
140                         //skriv noget med den vælger billede
```

```
141                 }
142                 t.Contents = contens;
143             }
144             tileSet.add(t);
145         }
146     }
147     gb.numbersOfCpus = map.getInt(5, 32) - 1;
148     System.out.println("numbersOfCpus " + (map.getInt(5, 32)));
149     gb.startGame(gb.numbersOfCpus, cpuPos);
150 
151     loadMapToBoat(map, "Player", 33, gb.player);
152     for (int i = 0; i < gb.NPCBoatArrayList.size() - 2; i++) {
153 
154         loadMapToBoat(map, "CPU", 34 + i, gb.NPCBoatArrayList.get(i));
155     }
156 
157 
158 }
159 
160 //adds shops to the map
161 void addShopANdPlayLocation(int numberOfTiles, ArrayList<Tile>
    tileSet, ArrayList<Integer> sandTiles, int numREmaningShops) {
162     int ypos = -1, xPos = -1;
163     for (int i = 0; i < tileSet.size(); ++i) {
164         Tile tile = tileSet.get(i);
165         for (int j = 0; j < sandTiles.size(); ++j) {
166 
167             if (sandTiles.get(j) == i) {
168 
169                 if (numREmaningShops < 0) {
170                     addShopANdPlayLocation(numberOfTiles, tileSet, sandTiles,
    numREmaningShops);
171                     break;
172 
173                 }
174                 if (tile.xPos > 2 && tile.xPos < 30 && tile.yPos > 2 && tile.
    yPos < 30) {
175                     int numbersOfWaterTiles = 0;
176                     numbersOfWaterTiles += checkIfLoactionHaveWater(tileSet, i
     + 1);
177                     numbersOfWaterTiles += checkIfLoactionHaveWater(tileSet, i
     + 35);
178                     numbersOfWaterTiles += checkIfLoactionHaveWater(tileSet, i
     + 36);
179                     numbersOfWaterTiles += checkIfLoactionHaveWater(tileSet, i
     + 37);
180                     numbersOfWaterTiles += checkIfLoactionHaveWater(tileSet, i
```

```
180   - 1);
181                   numbersOfWaterTiles += checkIfLoactionHaveWater(tileSet, i
      - 35);
182                   numbersOfWaterTiles += checkIfLoactionHaveWater(tileSet, i
      - 36);
183                   numbersOfWaterTiles += checkIfLoactionHaveWater(tileSet, i
      - 37);
184                   if (numberOfTiles > 1) {
185                      if (numberOfTiles > 1 &&
186                            numREmaningShops != 0
187                            && (ypos + 2 > tileSet.get(i).yPos && tileSet.get(i).
      xPos > xPos + 2)) {
188                         System.out.println("Shops X:" + tileSet.get(i).xPos + "
      Y:" + tileSet.get(i).yPos);
189                         Tile t = new ShopTile(p, "SHOP", tileSet.get(i).xPos,
      tileSet.get(i).yPos);
190                         tileSet.set(i, t);
191                         ypos = tileSet.get(i).yPos;
192                         xPos = tileSet.get(i).xPos;
193                         numREmaningShops--;
194                      } else {
195                         cpuPos.add(new PVector(tileSet.get(i).xPos, tileSet.get(i).
      yPos));
196                      }
197
198                   }
199                }
200
201                break;
202             }
203
204          }
205       }
206
207       if (numREmaningShops != 0) {
208          System.out.println("rs: " + numREmaningShops);
209          for (int j = 0; j < tileSet.size(); ++j) {
210             if (numREmaningShops == 0) {
211                break;
212             }
213             int i = (int) p.random(64, tileSet.size());
214             int numbersOfWaterTiles = 0;
215             numbersOfWaterTiles += checkIfLoactionHaveWater(tileSet, i + 1
      );
216             numbersOfWaterTiles += checkIfLoactionHaveWater(tileSet, i + 35
      );
217             numbersOfWaterTiles += checkIfLoactionHaveWater(tileSet, i + 36
```

```
217 );
218             numbersOfWaterTiles += checkIfLoactionHaveWater(tileSet, i + 37
    );
219             numbersOfWaterTiles += checkIfLoactionHaveWater(tileSet, i - 1);
220             numbersOfWaterTiles += checkIfLoactionHaveWater(tileSet, i - 35
    );
221             numbersOfWaterTiles += checkIfLoactionHaveWater(tileSet, i - 36
    );
222             numbersOfWaterTiles += checkIfLoactionHaveWater(tileSet, i - 37
    );
223             if (numbersOfWaterTiles > 1 && !tileSet.get(i).Contents.equals("
    BORDER")) {
224                 Tile t = new ShopTile(p, "SHOP", tileSet.get(i).xPos, tileSet.get
    (i).yPos);
225                 tileSet.set(i, t);
226                 numREmaningShops -= 1;
227                 System.out.println("rs: " + numREmaningShops);
228 
229             } else {
230             }
231         }
232     }
233 }
234 
235 //checks if j tile is water
236 int checkIfLoactionHaveWater(ArrayList<Tile> tileSet, int j) {
237     if (j < tileSet.size() && j > 0) {
238         if (tileSet.get(j).Contents.equals("WATER")) {
239             return 1;
240         } else {
241             return 0;
242         }
243     } else {
244         return 0;
245     }
246 }
247 
248 //This feature bothers the game
249 void generateGame(int numberOfTiles, ArrayList<Tile> tileSet) {
250 
251     ArrayList<Integer> sandTiles = new ArrayList<>();
252     ArrayList<PVector> waterTiles = new ArrayList<>();
253     int in = 0;
254     float xNoise = 0.0f;
255 
256     for (int x = -1; x < numberOfTiles + 3; ++x) {
257         xNoise += increment;
```

```
258             float yNoise = 0.0f;
259             for (int y = -1; y < numberOfTiles + 3; ++y) {
260                 yNoise += increment;
261                 float tileFloat = p.noise(xNoise, yNoise) * 255;
262                 Tile t = new TerrainTile(p, "", x, y);
263                 if ((x <= 0 || y <= 0) || (33 <= x || 33 <= y)) {
264                     t.Contents = "BORDER";
265                 } else {
266                     if (tileFloat > 99) {
267                         waterTiles.add(new PVector(x, y));
268                         t.Contents = "WATER";
269                     } else if (tileFloat > 85) {
270                         sandTiles.add(in);
271                         t.Contents = "SAND";
272                     } else {
273                         t.Contents = "GRASS";
274                     }
275                 }
276                 tileSet.add(t);
277                 in++;
278             }
279         }
280         addShopANdPlayLocation(numberOfTiles, tileSet, sandTiles, 3);
281         if (gb.cpuPos != null)
282             gb.cpuPos.clear();
283         gb.cpuPos = waterTiles;
284     }
285 }
286
287
```

```java
import processing.core.PApplet;

import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
import java.io.File;
import java.util.Scanner;

public class NomalButton extends Button {

    //---------- CONSTRUCTOR :) ----------\\
    NomalButton(PApplet papp, int posX, int posY, int sizeX, int sizeY, String text) {
        super(papp, posX, posY, sizeX, sizeY, text);
    }
    //----------METHODS----------\\

    //This function checks if the mouse is over the button and is clicked change it clicked to true
    @Override
    void registerClick(float mouseX, float mouseY) {
        this.mouseX = mouseX;
        this.mouseY = mouseY;
        if (mouseX > positionX &&
                mouseX < positionX + sizeX &&
                mouseY > positionY &&
                mouseY < positionY + sizeY) {
            try {
                File file = new File("Music/k.wav");
                Scanner scanner = new Scanner(System.in);
                AudioInputStream kas = AudioSystem.getAudioInputStream(file);
                Clip k = AudioSystem.getClip();
                k.open(kas);
                k.start();
            } catch (Exception e) {
                e.printStackTrace();
            }
            clicked = true;
        }
    }

    //This funktion set clicked to false
    void registrerRelease() {
        clicked = false;
    }
}

```

```
 1  import processing.core.PApplet;
 2  import processing.core.PSurface;
 3  import processing.core.PVector;
 4
 5  import javax.sound.sampled.FloatControl;
 6
 7  public class SettingMenu {
 8      //the variables
 9      PApplet p;
10      //menu variables
11      Boolean visible = false;
12      //determines how far you have come in the display resutlion lists
13      int displayResolutionInt = 1;
14      //last displayResolutionInt
15      int lastDisplayResolutionInt = 1;
16      //This variable helps to scale all ui elements with the screen size
17      public float size = 1;
18      //the list of sounds volume and what to type on the screen
19      PVector[] volumes ={new PVector(0,-80),new PVector(25,-20),new
        PVector(50,0),new PVector(75,3),new PVector(100,6)};
20      //this variable tells where on the volume list one has reached
21      int volInt =2;
22      //the list of the different screen sizes
23      PVector[] displayResolution = {new PVector(640, 360, 0.5f), new PVector(
        1280, 720, 1), new PVector(1600, 900, 1.25f), new PVector(1920, 1080, 1.5f
        )};
24      //quick access to the pause menu
25      PauseMenu pauseMenu;
26      //all text fields
27      TextField tfNumbersOfPlayers, tfMaxRound,tfGenNum;
28      //This one licks if everything has been resized
29      boolean backToMainMenu = true;
30      //All the buttons
31      NomalButton ResLeft, ResRight, backToMain,btnVolUp,btnVolDown;
32      //the size of the screen
33      int screenWidth, screenHeight;
34      //quick access to gameboard
35      GameBoard gb;
36      //quick access to savemanger
37      SaveManger sm;
38
39      //---------- CONSTRUCTOR :) ----------\\
40      SettingMenu(PApplet p) {
41          this.p = p;
42          btnVolDown = new NomalButton(p, 320, 100, 50, 50, "<");
43          btnVolUp = new NomalButton(p, 910, 100, 50, 50, ">");
44          ResLeft = new NomalButton(p, 320, 250, 50, 50, "<");
```

```
45          ResRight = new NomalButton(p, 910, 250, 50, 50, ">");
46          tfNumbersOfPlayers = new TextField(p, 200, 450, 200, 50, "Antal af
    modspiller");
47          tfNumbersOfPlayers.acceptLetters = false;
48          tfMaxRound = new TextField(p, 540, 450, 200, 50, "Antal Rundter");
49          tfMaxRound.acceptLetters = false;
50          tfGenNum = new TextField(p, p.width-400, 450, 200, 50, "Generation
    nummer");
51          tfGenNum.acceptLetters = false;
52          backToMain = new NomalButton(p, 540, 600, 200, 50, "Back to Menu"
    );
53      }
54      //----------METHODS----------\\
55
56      //This function draws the menu
57      void drawMenu() {
58          if (visible) {
59              p.image(main.bg,0,0,p.width,p.height);
60              p.textSize(32 * size);
61              p.text("Musikvolumen:", 320*size,70*size);
62              p.text("Skærmopløsning:", 320*size,230*size);
63              p.text("Spille Instillinger:", 320*size,410*size);
64              p.textSize(16 * size);
65              String displayInfo = (int) displayResolution[displayResolutionInt].x +
    " X " + (int) displayResolution[displayResolutionInt].y;
66              p.text(displayInfo,
67                      (p.width/2- p.textWidth(displayInfo) / 2) , (280) * size);
68              String volInfo = volumes[volInt].x +"%";
69              p.text(volInfo,
70                      (p.width/2- p.textWidth(displayInfo) / 2) , (130) * size);
71              p.textSize(16 * size);
72
73
74              btnVolUp.drawButton();
75              btnVolDown.drawButton();
76              backToMain.drawButton();
77              ResLeft.drawButton();
78              ResRight.drawButton();
79              tfMaxRound.drawField();
80              tfNumbersOfPlayers.drawField();
81              tfGenNum.drawField();
82
83              screenResManger();
84
85
86              if(btnVolUp.clicked){
87                  if(volInt +1 !=  volumes.length){
```

```
 88                 volInt += 1;
 89                 changeVolume(volumes[volInt].y);
 90              }
 91              btnVolUp.registrerRelease();
 92
 93           }
 94
 95           if(btnVolDown.clicked){
 96              if(volInt -1 !=  -1){
 97                 volInt -= 1;
 98                 changeVolume(volumes[volInt].y);
 99              }
100              btnVolDown.registrerRelease();
101
102           }
103
104
105        }
106     }
107
108     //This feature changes the volume of the game
109     void changeVolume(float volu){
110        FloatControl vol = (FloatControl) main.bgmusic.getControl(FloatControl
       .Type.MASTER_GAIN);
111        vol.setValue(volu);
112
113
114     }
115
116     //this function resizes the screen
117     void btnChangeScreen(MainMenu mm, GameBoard gb) {
118        if (backToMain.clicked) {
119
120           if (backToMainMenu) {
121              mm.reSizeMainMenu();
122              mm.visible = true;
123           } else {
124              pauseMenu.reSizePauseMenu();
125              pauseMenu.visible = true;
126           }
127           visible = false;
128           backToMain.registrerRelease();
129        }
130     }
131
132     //this function resizes the screen
133     void screenResManger() {
```

```
134         if (visible) {
135             if (ResLeft.isClicked()) {
136                 lastDisplayResolutionInt = displayResolutionInt;
137                 displayResolutionInt--;
138                 if (displayResolutionInt < 0)
139                     displayResolutionInt = displayResolution.length - 1;
140 
141                 if (displayResolutionInt == displayResolution.length - 1) {
142                     screenWidth = getFrame(p.getSurface()).getX();
143                     screenHeight = getFrame(p.getSurface()).getY();
144                     displayResolution[displayResolutionInt].x = screenWidth;
145                     displayResolution[displayResolutionInt].y = screenHeight;
146                     p.frame.setLocation(0, 0);
147                     p.frame.setSize(p.displayWidth, p.displayHeight);
148                 } else {
149                     p.frame.setLocation(screenWidth, screenHeight);
150                     p.frame.setSize((int) displayResolution[displayResolutionInt].x
        , (int) displayResolution[displayResolutionInt].y);
151                 }
152 
153                 size = displayResolution[displayResolutionInt].z;
154 
155                 reSizeMenu(size);
156                 System.out.println("size: " + size);
157                 ResLeft.registrerRelease();
158             }
159 
160             if (ResRight.isClicked()) {
161                 lastDisplayResolutionInt = displayResolutionInt;
162                 displayResolutionInt++;
163                 if (displayResolutionInt == displayResolution.length)
164                     displayResolutionInt = 0;
165 
166                 if (displayResolutionInt == displayResolution.length - 1) {
167                     screenWidth = getFrame(p.getSurface()).getX();
168                     screenHeight = getFrame(p.getSurface()).getY();
169                     p.frame.setLocation(0, 0);
170                     p.frame.setSize(p.displayWidth, p.displayHeight);
171                 } else {
172                     p.frame.setLocation(screenWidth, screenHeight);
173                     p.frame.setSize((int) displayResolution[displayResolutionInt].x
        , (int) displayResolution[displayResolutionInt].y);
174                 }
175 
176                 size = displayResolution[displayResolutionInt].z;
177 
178 
```

```
179             // RestSettings();
180             reSizeMenu(size);
181             System.out.println("size: " + size);
182             ResRight.registrerRelease();
183
184         }
185
186
187     }
188 }
189
190 //this feature adjusts the size of your screen
191 public static final javax.swing.JFrame getFrame(final PSurface surface) {
192     return (javax.swing.JFrame) ((processing.awt.PSurfaceAWT.
    SmoothCanvas) surface.getNative()).getFrame();
193 }
194
195 //This function for all buttons and look to fit with the screen size
196 void reSizeMenu(float s) {
197     pauseMenu.gb.scaleSize = s;
198     pauseMenu.mainMenu.scaleSize = s;
199     reSizeBtn(s, ResLeft);
200     reSizeBtn(s, ResRight);
201     reSizeBtn(s, backToMain);
202     reSizeFT(s, tfNumbersOfPlayers);
203     reSizeFT(s, tfMaxRound);
204     reSizeFT(s,tfGenNum);
205     reSizeBtn(s,btnVolDown);
206     reSizeBtn(s,btnVolUp);
207
208 }
209
210 //this function resizes the buttons
211 public void reSizeBtn(float s, Button btn) {
212     btn.size = s;
213
214 }
215 //this function resizes the textfeild
216 public void reSizeFT(float s, TextField tf) {
217     tf.size = s;
218
219 }
220
221 //This function allows you to click on the menu
222 void menuMouseClick() {
223     if (visible) {
224         tfGenNum.registerClick(p.mouseX, p.mouseY);
```

```
225             ResLeft.registerClick(p.mouseX, p.mouseY);
226             ResRight.registerClick(p.mouseX, p.mouseY);
227             backToMain.registerClick(p.mouseX, p.mouseY);
228             tfNumbersOfPlayers.registerClick(p.mouseX, p.mouseY);
229             tfMaxRound.registerClick(p.mouseX, p.mouseY);
230             btnVolDown.registerClick(p.mouseX, p.mouseY);
231             btnVolUp.registerClick(p.mouseX, p.mouseY);
232         }
233     }
234     //this feature lets you type in the console
235     void menuKeyTyped() {
236         if (visible) {
237             tfGenNum.keyinput(p.key);
238             tfNumbersOfPlayers.keyinput(p.key);
239             tfMaxRound.keyinput(p.key);
240 
241             if (tfGenNum.input.length() > 0) {
242                 sm.increment = Float.valueOf(tfGenNum.input)/100;
243             }
244             if (tfNumbersOfPlayers.input.length() > 0) {
245                 gb.numbersOfCpus = Integer.valueOf(tfNumbersOfPlayers.input);
246             }
247             if (tfMaxRound.input.length() > 0) {
248                 gb.maxRounds = Integer.valueOf(tfMaxRound.input);
249             }
250 
251         }
252     }
253 }
254 
```

```java
import processing.core.PApplet;
import processing.core.PImage;

public class TerrainTile extends Tile {

    //---------- CONSTRUCTOR :) ----------\\
    public TerrainTile(PApplet p, String C, float x, float y) {

        super(p, C, x, y);

    }
    //----------METHODS----------\\

    @Override
    public void drawShopMenu(Player player, float scaleSize) { }

    @Override
    public void showShop(float s) { }

    @Override
    public void clickShop() { }

    //this function change the image on the tile. depends on the type of terran it is
    @Override
    public void setTileImage() {
        if (Contents.equals("WATER")) {
            PImage tile;
            tile = p.loadImage("w" + 1 + ".png");
            tileImage = tile;
        } else if (Contents.equals("GRASS")) {
            PImage tile;
            tile = p.loadImage("g" + (int) p.random(1, 5) + ".png");
            tileImage = tile;

        } else if (Contents.equals("SAND")) {
            PImage tile;
            tile = p.loadImage("s" + 1 + ".png");
            tileImage = tile;
        } else if (Contents.equals("BORDER")) {
            tileImage = p.loadImage("border.png");
        }
    }
}

```

```
 1  import processing.core.PApplet;
 2
 3  public class ChooseGameMenu {
 4      //Menu variables
 5      PApplet p;
 6      //This is so the menu can easily access the game board
 7      GameBoard gb;
 8      //Quick access to the main menu
 9      MainMenu mainMenu;
10
11      //About the menu can be seen
12      Boolean visible = false;
13
14      //This variable is only used if you have completed a game.
15      // Then we would like the users to click on the
16      // button that is used to make a new game and not
17      // the one that opens the current game. therefore we hide it away
18      Boolean needNewGame = false;
19
20      //All menu buttons
21      NomalButton btnPlay, btnLoadGame, btnNewGame, btnBackToMenu;
22
23      //---------- CONSTRUCTOR :) ----------\\
24      ChooseGameMenu(PApplet p, GameBoard gb, MainMenu mainMenu) {
25          this.p = p;
26          this.gb = gb;
27          this.mainMenu = mainMenu;
28          btnPlay = new NomalButton(p, 640 - 160, 100, 320, 50, "Forsæt spil");
29          btnLoadGame = new NomalButton(p, 640 - 160, 160, 320, 50, "Indlæs
    Spil");
30          btnNewGame = new NomalButton(p, 640 - 160, 220, 320, 50, "Nyt Spil"
    );//280
31          btnBackToMenu = new NomalButton(p, 640 - 160, 280, 320, 50, "
    Tilbage");
32
33
34      }
35
36      //----------METHODS----------\\
37
38      //draws the menu
39      void drawMenu() {
40          p.clear();
41          p.image(main.bg,0,0,p.width,p.height);
42          if(!needNewGame)
43          btnPlay.drawButton();
44
```

```
45
46          btnLoadGame.drawButton();
47          btnNewGame.drawButton();
48          btnBackToMenu.drawButton();
49          if (btnPlay.clicked) {
50
51              gb.scaleSize = mainMenu.scaleSize;
52              gb.reSizeGamebord();
53              gb.visible = true;
54              needNewGame =false;
55              mainMenu.visible = false;
56              visible = false;
57              btnPlay.registrerRelease();
58          }
59
60          if (btnLoadGame.clicked) {
61              needNewGame =false;
62              System.out.println("Bib bab og andre robotlyde");
63              p.selectInput("Select a file to process:", "loadedMap");
64              btnLoadGame.registrerRelease();
65          }
66
67          if (btnNewGame.clicked) {
68              needNewGame =false;
69              gb.scaleSize = mainMenu.scaleSize;
70              gb.reSizeGamebord();
71              gb.tileSet.clear();
72              gb.saveManger.generateGame(32, gb.tileSet);
73              gb.startGame(gb.numbersOfCpus, gb.saveManger.cpuPos);
74              gb.visible = true;
75              gb.roundCount = 0;
76              gb.turnCount = 0;
77              mainMenu.visible = false;
78              visible = false;
79              needNewGame = true;
80              btnNewGame.registrerRelease();
81          }
82
83          if (btnBackToMenu.clicked) {
84              mainMenu.visible = true;
85              visible = false;
86              btnBackToMenu.registrerRelease();
87          }
88
89
90      }
91  }
```