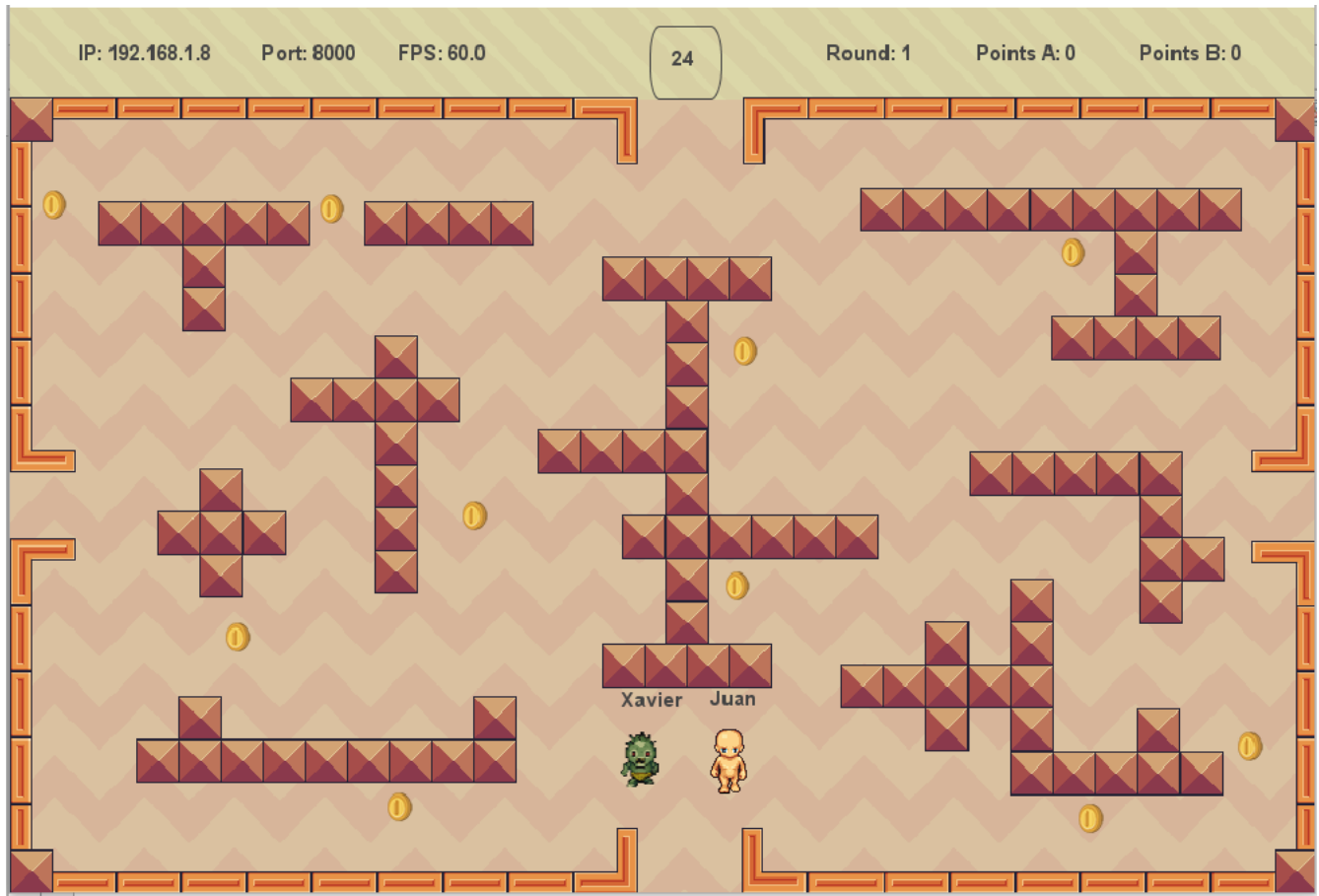


Zombie Game Documentació



Nom: Xavier Lliteras, Juan Antonio Bieta
Curs: DAM2A.

Index

Introducció.....	3
Normes i funcionament del joc.....	3
Part gràfica.....	5
Implementació del projecte.....	8
Diagrama de classes UML.....	8
Classes i explicació del seu funcionament.....	8
ZombieGame.....	8
Pad.....	13
VisibleObject:.....	14
Alive.....	14
Controlled:.....	15
Zombie:.....	16
Human:.....	16
Static:.....	17
Wall:.....	17
Obstacle:.....	18
Coin:.....	18
Rules:.....	19
Looby:.....	20
ScoreBoard:.....	21
Comandament aplicació mòbil:.....	21
JoyStickActivity.....	21
MainActivity.....	21
JoyStickClass.....	21
Conexions.....	21
Server.....	22
ConnectionHandler.....	23
ZombieGame.....	23
Pad.....	25
Webgrafia.....	26

Introducció

Es proposa la relització d'un joc gràfic en java multijugador no online (una sola pantalla), en 2d des d'una vista de planta. Els jugadors es connectaran al joc a través d'un comandament que tindrà instal·lat al mòbil.

El joc consistirà bàsicament en la competició entre dos equips, un atacant i l'altre defensor, els atacants han d'intentar menjar-se als defensors i els defensors han d'escapar i intentar agafar monedes que sumaran punts, després d'un temps els defensors passaran a ser atacants i els atacants a ser defensors. Qui guanyi més punts al final del joc guanya la partida.

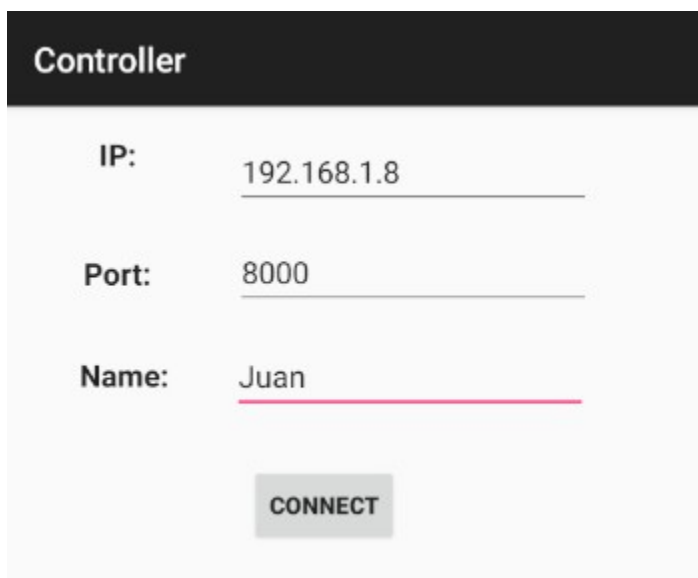
La temàtica del joc serà zombies contra humans amb animacions pròpies de cada grup.

Normes i funcionament del joc

Funcionament del joc.

Sempre es connectarà al joc mitjançant el comandament que tindrem instal·lat al mòbil. A l'aplicació li haurem d'indicar la IP i el port on tenim el joc en execució.

Una vegada connectats la nostra IP i nom apareixeran al Looby de l'aplicació des d'on podrem iniciar la partida.



The screenshot shows a mobile application interface with a dark header labeled "Controller". Below the header, there are three input fields: "IP:" with the value "192.168.1.8", "Port:" with the value "8000", and "Name:" with the value "Juan". The "Name:" field has a red underline. At the bottom, there is a grey button labeled "CONNECT".

Captura 1: Pantalla connexió. (Aplicació mòbil).

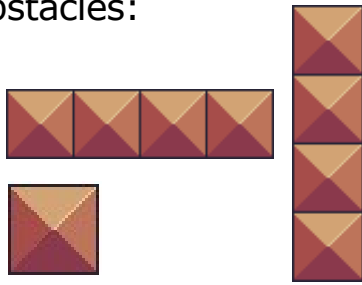
Una vegada ens hàgem connectat, no podrem triar l'equip que volem, se'ns assignarà automàticament, ja que d'aquesta manera no tindrem el risc de fer equips molt desiguals en nombre de jugadors.

Normes:

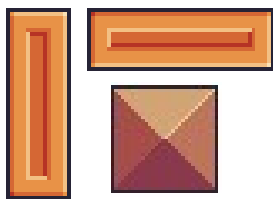
- Els jugadors apareixeran aleatòriament sense tocar-se entre equips o amb els obstacles.
- Els jugadors no podran sobrepassar els obstacles del mapa, en cas d'intentar sobrepassar un obstacle xocaran amb ell i aturaran tota la seva velocitat.
- Els atacants tenen l'habilitat d'atacar als defensors mitjançant la teletransportació quan. Els defensors tenen l'habilitat d'augmentar la seva velocitat de forma temporal per escapar d'un atac.
- A la part central dels extrems del mapa hi ha zones on es possible teletransportar-se a l'extrem contrari del mapa.
- Els defensors poden agafar les monedes que hi ha repartides pel mapa per a poder aconseguir punts, això ho faran mentre son perseguits.
- El temps per ronda es un minut.
- Quan els defensors hagin agafat totes les monedes del mapa, s'hagi acabat el temps o els atacants s'hagin menjat a tots els defensors es passarà de ronda o s'acabarà la partida en cas d'estar a l'última ronda.
- El mapa tindrà 10 monedes com a màxim i es repartirán de manera aleatòria. En fer canvi de ronda es reposaran les que falten.
- El nombre de jugadors mínim per començar una partida són dos.

Part gràfica

Obstacles:



Limits del mapa:



Fons:



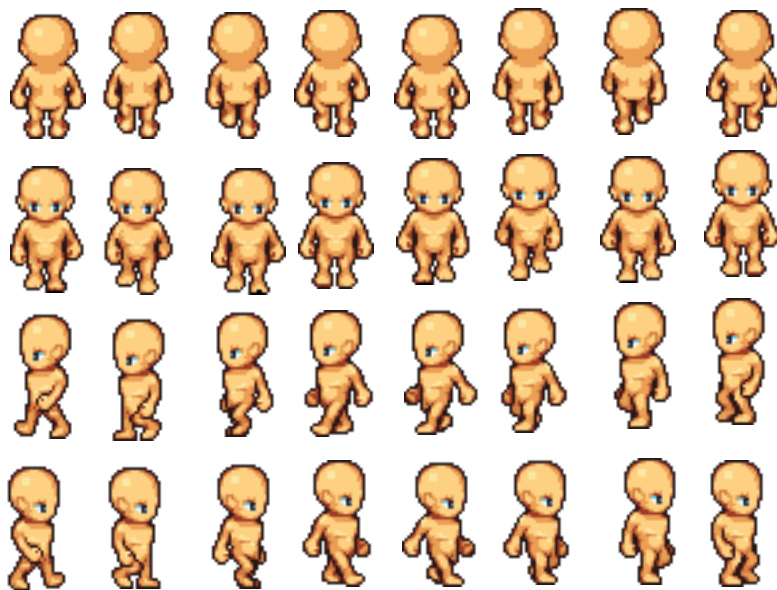
Consumibles:



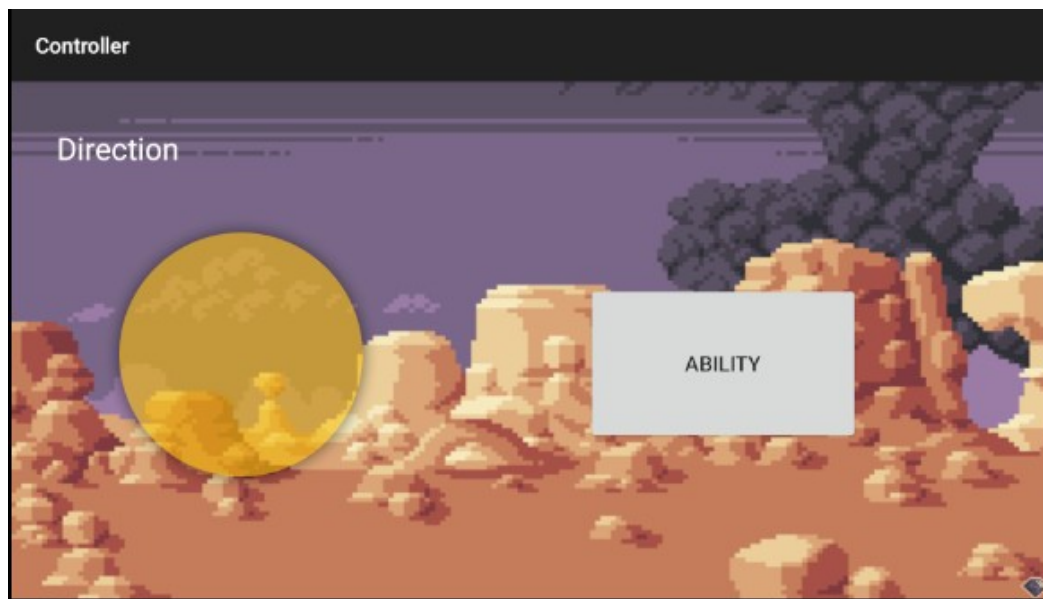
Zombies I animació de caminar.



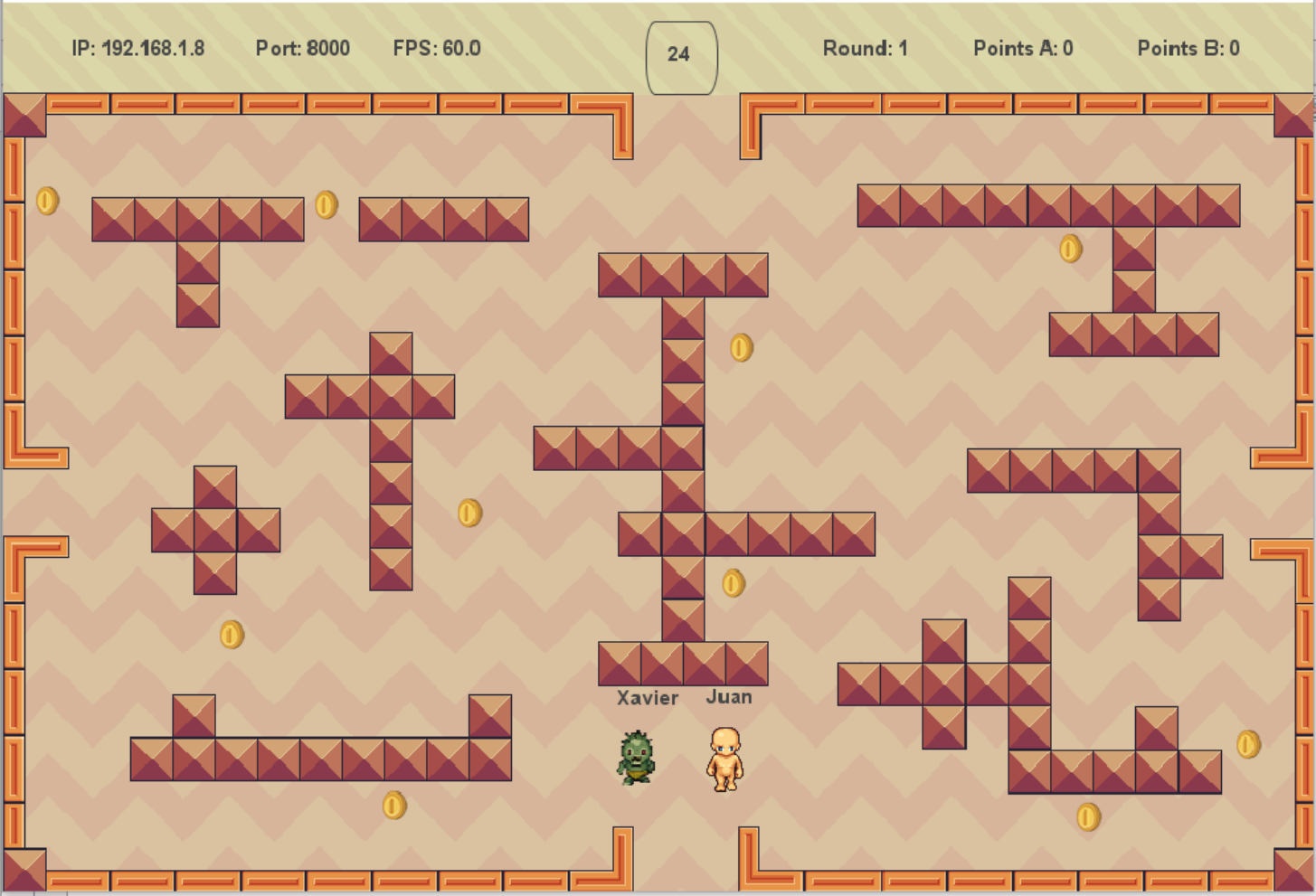
Humans i animació de caminar:



Interfície comandament mòbil:



Interficie general del joc (amb tots els elements anteriors):



Implementació del projecte

Diagrama de classes UML.

Enllaç Google Drive: <https://bit.ly/37n59ZH>.

Classes i explicació del seu funcionament.

ZombieGame

Aquí es on es du a terme la partida, es creen i gestionen tots els objectes visuals, també es defineix les seves interaccions. Es la master class. Hereda de JFrame.

Mètodes:

- Constructor: A n'aquest mètode es creen tots els components que formaran part d'aquesta classe (Mandos, Visualitzador Canvas, Marcadors, Normes, Looby).

També es defineixen les característiques de la finestra i es crea aquesta.

```
40 public ZombieGame() {  
41     super("ZombieGame");  
42     this.inicioPartida = false;  
43     this.pads = new ArrayList();  
44     this.visibleObjects = new ArrayList();  
45     this.server = new Server(this);  
46     this.score = new ScoreBoard();  
47     this.viewer = new Viewer(this);  
48     this.rules = new Rules(this);  
49     this.looby = new Looby(this);  
50     setSize(997, 701);  
51     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
52     setResizable(false);  
53     setVisible(true);  
54 }
```

Captura 2: Constructor de la classe ZombieGame.

- `getLocalIp`: Retorna un string amb la IP de l'equip, aquest mètode s'utilitzarà quan necessitem ensenyar la ip per pantalla.
- `prepareGame`: Afegeix el `Jpanel Looby` al `JFrame` de la classe i inicia el thread del servidor. Aquest mètode es executat immediatament després de crear la finestra.
- `getReady`: Crea els objectes visuals i afegeix el canvas al `JFrame` per a començar una partida, també inicia el canvas i reinicia els marcadors.
- `play`: Inicia i gestiona la partida durant tot el temps que duri. S'executa després de "`getReady`".
- `createVisibleObjects`: Executa altres mètodes que creen i afegeixen a l'array de objectes els objectes visuals de la partida (Parets, personatges, obstacles, monedes). S'utilitza a l'inici de cada partida.
- `blockMoviments`: Bloquetja tots els moviments perquè l'usuari no pugui moure el seu personatge. S'utilitza durant el temps de looby i a l'inici de cada ronda.
- `checkEndRound`: Comprova si la ronda s'ha acabat. La ronda s'acaba quan no queden humans, no queden monedes o s'ha acabat el temps.
- `endGame`: Assigna el `Jpanel del Looby` al `JFrame` de la propia classe i prepara tot per acabar una partida. s'executa després de acabar cada partida.
- `eliminateObjects`: Destruïx tots els threads de tots els objectes visuals i els elimina. Es necessari executar-lo després de acabar cada partida.
- `sleep`: Reb per paràmetre un nombre de segons i espera el temps indicat abans de seguir amb l'execució. S'utilitza a dins la partida perquè el marcador de segons baixi a velocitat de un segon per segon.

- changeCharacter: Incrementa la ronda i canvia els zombies per humans i els humans per zombies. s'executa al final de cada ronda.
- changeHuman: Canvia un huma per zombie. S'executa dins "changeCharacter".
- changeZombie: Canvia unzombie per huma. S'executa dins "changeCharacter".
- changePadDisconnected: Comprova que no fa més de 17 segons que el Pad s'ha desconectat, en cas contrari elimina el personatge assignat al Pad i també elimina el Pad. S'executa a dins el Pad quan la desconexió es evident.
- createCoins: Crea les monedes a llocs aleatoris del mapa. Es utilitza a l'inici de cada ronda.
- knowNumberCoins: Retorna el nombre de monedes que tenim al mapa actualment. S'utilitza per saber quantes en falten.
- removeCoins: Destruïx totes les monedes del mapa. S'utilitza a l'inici de cada ronda.
- checkCoins: Rep per parametre una posició (x,y) I retorna l'existència de una moneda a n'aquesta posició. S'utilitza quan es creen les monedes per comprovar que on volem crear una moneda no n'hi ha una.
- createCharacters: Crea de manera aleatoria el primer personatge de la partida, els altres els va creant a partir d'aquests respectant la paridad zombie humà. S'utilitza a l'inici de cada partida.
- createWalls: Crea totes les parets que fan de límit del mapa i les afegeix a l'arrayList de objectes visuals. S'utilitza a l'inici de cada partida.

- createObstacles: Crea tots els obstacles que no són límits del mapa. S'utilitza a l'inici de cada partida.
- createZombie: Crea un zombie.
- createHuman: Crea un humà.
- createPad: Reb per paràmetre un socket i crea un Pad o el reconecta en cas de ja existir. Si la partida ja ha començat no es podrà crear. S'utilitza quan un pad es conecta al servidor.
- addPlayerToLooby: Reb per paràmetre la ip i el nom del nou player I l'afegeix a la llista del looby. S'utilitza quan un Pad es connectat satisfactòriament.
- searchPad: Reb per paràmetre la id del Pad i retorna el Pad en cas d'existir, sino retorna null.
- searchControlled: Reb per paràmetre la id de un personatge, el cerca i si el troba el retorna sino retorna null.
- disconnectPad: Reb per paràmetre la id del Pad I elimina el Pad I el seu personatge. S'utilitza quan fa mes de 17 segons que el joc no reb resposta del pad.
- moveControlled: Reb per paràmetre la id del personatge que volem moure i el moviment que volem dur a terme, el mètode s'encarrega de trobar el jugador i executar l'ordre. S'executa en el Pad quan li enviam un moviment.
- colisionTestBorders: Comprova que no haguem colisionat amb els cantons del mapa. En cas contrari executa una ordre de les Rules per a teletransportar el personatge on correspongui. S'utilitza a dins el thread dels jugadors.
- colisionTestX: Reb per paràmetre un objecte visual i Comprova que no haguem colisionat de forma horitzontal amb cap altre objecte visual. S'utilitza per a desactivar la velocitat de x.

- colisionTestY: Reb per paràmetre un objecte visual i comprova que no haguem colisionat de forma vertical amb cap objecte visual. S'utilitza per a desactivar la velocitat de y.
- bordersColision: Retorna un integer amb un codi de colisió si ens passam de uns píxels.
- destroyHuman: Acaba el thread de un huma i l'elimina. S'utilitza en les colisions entre humans i zombies.
- colisionTeleport: En el cas del zombie comprova que no hi hagi cap objecte visual en el lloc on ens volem teletransportar. S'utilitza cada vegada que el zombie utilitza la seva habilitat.
- collectCoin: Reb per paràmetre un objecte visual que realment es una Coin l'elimina i suma als marcadors.
- sendObjectToLeft: Envia un objecte a la l'esquerra de la pantalla. S'utilitza en cas de sobrepassar el límit de de la dreta.
- sendObjectToRight: Envia un objecte a la dreta de la pantalla. S'utilitza en cas de sobrepassar el límit de l'esquerra.
- sendObjectToBottom: Envia un objecte a la part inferior de la pantalla. S'utilitza en cas de sobrepassar el límit d'adalt.
- sendObjectToTop: Envia un objecte a la part superior de la pantalla. S'utilitza en cas de sobrepassar el límit d'abaix.

Pad

Aquesta classe es la que gestiona la connexió i comunicació entre la aplicació mòbil i l'aplicació de escriptori. La classe implementa Runnable.

```
37 public Pad(Socket clientSock, ZombieGame GamePad) {  
38     this.clientSocket = clientSock;  
39     this.gamePad = GamePad;  
40     this.connected = true;  
41     this.localIp =  
42     this.clientSocket.getLocalAddress().getHostAddress();  
43     this.id =  
44     this.clientSocket.getInetAddress().getHostAddress();  
45     this.timeOut = System.nanoTime() / 1000000000;  
46     this.blockCharacter=true;  
47 }
```

Capura 3: Constructor de la classe Pad.

Mètodes de la classe:

- Constructor: S'encarrega de assignar el socket que se li envia per paràmetre al socket de la propia classe. També extreu la ip del socket i calcula un temps de timeOut per a comprovar una futura desconexió.
- ProccessClient: Es el mètode principal de la classe, s'encarrega de comprovar que no s'ha desconectat el comandament mitjançant missatges periodics i reb els missatges de l'aplicació mòbil. S'utilitza quan s'inicia el thread del comandament.
- reconectarMando: Rep per paràmetre un socket i l'assigna al de la propia classe. Reinicia el temps de desconexió.
- comprobarDesconexion: Comprova que no fa més de quatre segons que s'ha enviat l'últim missatge. S'executa tot el temps.
- decideMessage: Decideix que fer amb el tipus de missatge que reb de l'aplicació.

- processMessage: Es un filtre que va abans de decideMessage per a comprovar si ens envien un nom de jugador.

VisibleObject:

Aquesta classe conté l'estructura de tots els objectes del joc i s'utilitzara com herencia en les classes Alive i Static.

```
27 public VisibleObject(double x, double y, boolean state,  
28     ZombieGame zombieGameObjects,String id,int width, int height){  
29  
30     this.zombieGameObjects=zombieGameObjects;  
31     this.x=x;  
32     this.y=y;  
33     this.state=state;  
34     this.id=id;  
35     this.width=width;  
36     this.height=height;  
37 }
```

Captura 4: Constructor de la classe VisibleObject.

Alive

Aquesta classe conté l'estructura dels objectes que están vius del joc i s'utilitzara com herencia en la classe Controlled.

```
16 public Alive(double x, double y, boolean state,  
17     ZombieGame zombieGameObjects, String id, int width,  
18     int height,double velocityX,double velocityY) {  
19  
20     super(x, y, state, zombieGameObjects, id, width, height);  
21     this.velocityX=velocityX;  
22     this.velocityY=velocityY;  
23 }
```

Captura 5: Constructor de la classe Alive.

Controlled:

Aquesta classe conté l'estructura dels objectes controlats pel jugador, s'utilitzara com herencia en la classe Zombie i Human.

```
25 public Controlled(double x, double y, boolean state,  
26     ZombieGame zombieGameObjects, String id, int width,  
27     int height, double velocityX, double velocityY, double v,  
28     int animation, String name) {  
29     super(x, y, state, zombieGameObjects, id, width, height, velocityX, velocityY);  
30     this.actualFrame = 0;  
31     this.frameCounter = 0;  
32     this.moviments = new int[8];  
33     this.animation = animation;  
34     this.v = v;  
35     this.name = name;  
36     this.blockedAbility = false;  
37     this.blockedTimeOut = System.currentTimeMillis() / 1000000000;  
38 }
```

Captura 6: Constructor de la classe Controlled.

Mètodes:

- unBlockAbility: Desbloquetja l'abilitat al cap d'uns segons, per poder tornar a utilitzar-la.
- ColisionX i ColisionY: Retorna true o false si n'hi ha hagut alguna col·lisió.
- updateFrames: Actualitza el frame actual.
- isMoving: Comprova si el jugador s'està movent.
- setDirection: Canvia de direcció comprovant quin moviment està actualment utilitzantse.
- move: Comprova que el jugador no col·lisió, després es mou i actualitza els frames.

Zombie:

Aquesta classe conté els mètodes del zombie.

```
25 public Zombie(double x, double y, boolean state,  
26 ZombieGame zombieGameObjects, String id, int width,  
27 int height, double velocityX, double velocityY,  
28 double v, int animation, String name) {  
29     super(x, y, state, zombieGameObjects, id, width,  
30     height, velocityX, velocityY, v, animation, name);  
31     loadZombieFrames();  
32 }
```

Captura 7: Constructor de la classe Zombie.

Mètodes:

- ability: el zombie es teletransporta a una distancia determinada, comprovant primer que no n'hi hagi col·lisió amb els obstacles.
- checkColisionTeleport: comprova que n'hi hagi col·lisió amb obstacles abans d'utilitzar l'abilitat.
- unblockAbility: desbloqueja l'abilitat després d'haver estat utilitzada.
- loadZombieFrames: carga el frame necessari.
- render: pinta al zombie en el joc.

Human:

Aquesta classe conté els mètodes de l'humà.

```
24 public Human(double x, double y, boolean state,  
25             ZombieGame zombieGameObjects, String id, int width,  
26             int height, double velocityX, double velocityY, double v,  
27             int animation, String name) {  
28     super(x, y, state, zombieGameObjects, id, width,  
29           height, velocityX, velocityY, v, animation, name);  
30     loadHumanFrames();  
31 }
```

Captura 8: Constructor de la classe Human.

Mètodes:

- ability: augmenta la velocitat del jugador.
- checkAbility: comprova si s'està utilitzant l'abilitat.
- loadHumanFrames: carga el frame necessari.
- render: pinta al zombie en el joc.

Static:

Aquesta classe conté l'estructura dels objectes estatics del joc, d'ella hereden els obstacles i les monedes.

```
14 public Static(double x, double y, boolean state,  
15             ZombieGame zombieGameObjects, String id,  
16             int selectObject, int width, int height) {  
17     super(x, y, state, zombieGameObjects, id, width, height);  
18     this.selectObject=selectObject;  
19 }
```

Captura 9: Constructor de la classe Static.

Wall:

Aquesta classe conté els mètodes de les parets.

```
22 public Wall(double x, double y, boolean state,  
23             ZombieGame zombieGameObjects, String id,  
24             int selectObject, int width, int height) {  
25     super(x, y, state, zombieGameObjects,  
26           id, selectObject, width, height);  
27     loadObstacles();  
28 }
```

Captura 10: Constructor de la classe Wall.

Mètodes:

- loadObstacles: carga l'imatge de l'obstacle a partir del nombre assignat a l'objecte.
- render: pinta l'objecte en el joc.

Obstacle:

Aquesta classe conté els mètodes dels obstacles.

```
22 public Obstacle(double x, double y, boolean state,  
23                  ZombieGame zombieGameObjects, String id,  
24                  int selectObject, int width, int height) {  
25     super(x, y, state, zombieGameObjects,  
26           id, selectObject, width, height);  
27     loadObstacles();  
28 }
```

Captura 11: Constructor de la classe Obstacle.

Mètodes:

- loadObstacles: carga l'imatge de l'obstacle a partir del nombre assignat a l'objecte.
- render: pinta l'objecte en el joc.

Coin:

Aquesta classe conté els mètodes de la moneda.

```
25     public Coin(double x, double y, boolean state,
26     ZombieGame zombieGameObjects, String id,
27     int selectObject, int width, int height) {
28         super(x, y, state, zombieGameObjects,
29         id, selectObject, width, height);
30         this.frameCounter=0;
31         this.actualFrame=0;
32         loadCoin();
33     }
```

Captura 12: Constructor de la classe Coin.

Mètodes:

- loadCoin: carga l'imatge de la moneda, i fa l'animació canviant els frames.
- render: pinta l'objecte en el joc.

Rules:

Aquesta classe s'encarga de dictar les regles del joc, que passa quan dos objectes col·lisionen.

```
15     public Rules(ZombieGame zombieGmaeRules){
16         this.zombieGameRules=zombieGmaeRules;
17     }
```

Captur 13: Constructor de la classe Rules.

Mètodes:

- processCollision: Comprova quins objectes han col·lisionat. Si ha sigut un humà i un zombie, el humà mor, si ha sigut un humà i una moneda el humà la recogeix.
- collisionBorders: mou al jugador si col·lisiona amb un marge.

Looby:

Aquesta classe conté els mètodes que creen el lobby del joc.

```
45 public Looby(ZombieGame zombieGameLobby) {
46     this.zombieGameLobby = zombieGameLobby;
47     setBorder(BorderFactory.createTitledBorder(BorderFactory.createEtchedBorder()
48     , "Looby", TitledBorder.CENTER, TitledBorder.TOP));
49     setLayout(new GridBagLayout());
50     this.modeloTabla = new DefaultTableModel() {
51         @Override
52         public boolean isCellEditable(int row, int column) {
53             return false;
54         }
55     };
56     this.tablaLooby = new JTable(this.modeloTabla) {
57         @Override
58         public Dimension getPreferredSize() {
59             return new Dimension(600, 400);
60         }
61     };
62     this.modeloTabla.addColumn("Nombre");
63     this.modeloTabla.addColumn("IP");
64     this.nombresJugadores = new HashMap<String, String>();
65     this.errorMessage = new JLabel("");
66     setBackground("img/fondo2.png");
67     configurarVentana();
68 }
```

Captura 13: Constructor de la classe Looby.

Mètodes:

- configurarVentana: Es creen tots els JPanel, JButton i JLabel necessaris per la creació del lobby.
- configurarConstraints: Mètode per ajustar el tamany i posició dels elements del JPanel.
- addPlayer: afegeix un jugador al JTable, amb el seu nom i ip.
- removePlayer: Lleva un jugador del Jtable.
- updatePanel: Actualitza l'informació del JTable.
- cleanJTable: Limpia el Jtable.

- `paintComponent`: pinta el fons del `JPanel` i el redimensiona si es necessari.
- `actionPerformed`: Comprova si n'hi ha suficients jugadors per començar la partida.

ScoreBoard:

Aquesta classe s'encarga de actualitzar el marcador de la partida i controlar el temps de cada ronda, així com canviarla.

```

18 public ScoreBoard() {
19     this.puntosEquipoA=0;
20     this.puntosEquipoB=0;
21     this.reloj=61;
22     this.ronda=1;
23 }

```

Captura 14: Constructor de la classe ScoreBoard .

Comandament aplicació mòbil:

Activities:

JoyStickActivity

S'encarrega de enviar missatges a l'aplicació quan l'usuari mou el joystick, també envia missatges periodics a l'aplicació per a saber si es segueix connectat, això ho fa mitjançant un thread que li permet rebre missatges i enviar-los.

MainActivity

S'encarrega de realitzar la connexió entre el mòbil i l'aplicació de escriptori.

JoyStickClass

S'encarrega de la lògica matemàtica del joystick i el seu funcionament.

Conexions

Server

Per a connectar-nos a l'aplicació primer necessitam tenir un servidor que pugui rebre diferents peticions. En el nostre cas es una classe que es diu Server i quan reb una petició de connexió la deriva a un thread apart per a poder rebre més peticions sense que el socket quedi ocupat.

```
@Override
public void run() {
    ServerSocket serverSock = configurePort();
    System.out.println("Waiting for a client...");
    System.out.println("Port: "+serverSock.getLocalPort());
    while (true) {
        try {
            this.clientSock = serverSock.accept();
            this.cliAddr =
                clientSock.getInetAddress().getHostAddress();
            Thread t =
                new Thread(new ConnectionHandler(this.clientSock,
                    this.cliAddr, this.port, this.zombieGameServer));
            t.start();
        } catch (IOException ex) {
            Logger.getLogger(Server.class.getName()).
                log(Level.SEVERE, null, ex);
        }
    }
}
```

Captura 15: Mètode Run de la classe Server.

ConnectionHandler

Una vegada que la petició ha estat derivada a un objecte de tipus "ConnectionHandler" es determina el missatge que l'aplicació mòbil ens ha enviat, hi ha dues opcions:

- bye: Desconecta el pad.
- cntpad: Conecta el pad a través de la classe ZombieGame.

```
private void doRequest(String line) throws IOException {
    System.out.println(line);
    switch (line) {
        case "cntpad":
            this.zombieGameConnection.
                createPad(this.clientSocket);
            System.out.println("Pad detected");
            break;
        case "bye":
            this.zombieGameConnection.
                disconnectPad(
                    this.clientSocket.getInetAddress().
                        getHostAddress());
            break;
        default:
            System.out.println("Ignoring input line");
            this.clientSocket.close();
            break;
    }
}
```

Captura 16: Opcions classe ConnectionHandler

ZombieGame

En cas de que el pad ens hagi enviat un missatge pera conectar-se, es crearà un nou Pad a la classe ZombieGame I s'iniciara el seu thread, també pot passar que el pad ja existís i ara es reconecti.

```
public void createPad(Socket clientSock) {
    Pad pad = searchPad(clientSock.getInetAddress().
        getHostAddress());
    if (pad == null) {
        if (!this.inicioPartida) {
            pad = new Pad(clientSock, this);
            this.pads.add(pad);
            (new Thread(pad)).start();
        }
    } else {
        pad.reconectarMando(clientSock);
        (new Thread(pad)).start();
    }
}
```

Captura 17: Creació de Pad a la classe ZombieGame.

Si la partida ja s'hagués iniciat no crearia el nou pad però si el podria reconectar.

Pad

Una vegada creat el Pad s'enviarà un missatge a l'aplicació mòbil per a comprovar si segueix connectada, si l'aplicació respon el temps de resposta es reiniciarà, en cas contrari seguirà augmentant, si arriba a quatre segons el Pad es desconectarà.

```
private void processClient() {
    String line;
    this.out.println("como estas");
    while (this.connected) {
        comprobarDesconexion();
        try {
            if (this.in.ready()) {
                line = this.in.readLine();
                decideMessage(line);
            }
            try {
                Thread.sleep(5);
            } catch (InterruptedException ex) {
                System.out.println("App cerrada");
            }
        } catch (IOException ex) {
            Logger.getLogger(Pad.class.
                getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

Captura 18: Pad reb I envia missatges.

```
public void comprobarDesconexion() {
    if (System.nanoTime() / 1000000000 - this.timeOut > 4) {
        this.connected = false;
        this.gamePad.checkPadDisconnected(this.id);
    }
}
```

Captura 19: Pad comprova desconexió

Webgrafia

Enllaç del projecte a github: <https://github.com/Xavier192/ZombieGame>.