**11. Grafikus változat tervei**

5. – Pandora

Konzulens:

Elekes Márton

Csapattagok

|  |  |  |
| --- | --- | --- |
| Szabó András | GA9BCR | sbasiliscus@gmail.com |
| Katona Benedek | LNU506 | kbeniboy2004@gmail.com |
| Lipóczi Levente László | SUSXSW | lipoczi.levente@gmail.com |
| Fekete Botond | VZ6YKQ | botond.fekete.010@gmail.com |
| Kőműves Marcell | LLDPZI | kommarci03@gmail.com |

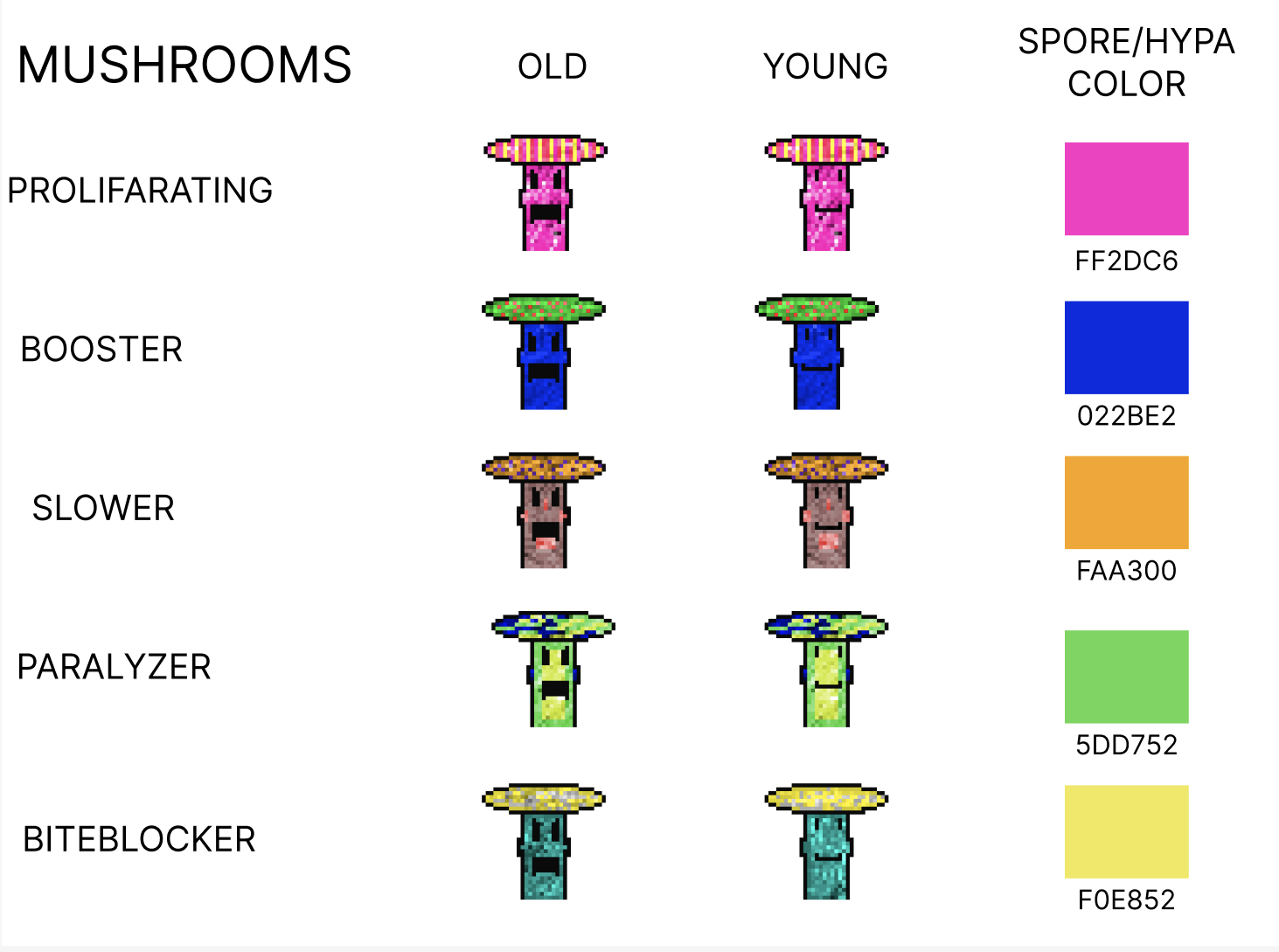
# **11.** **Grafikus felület specifikációja**

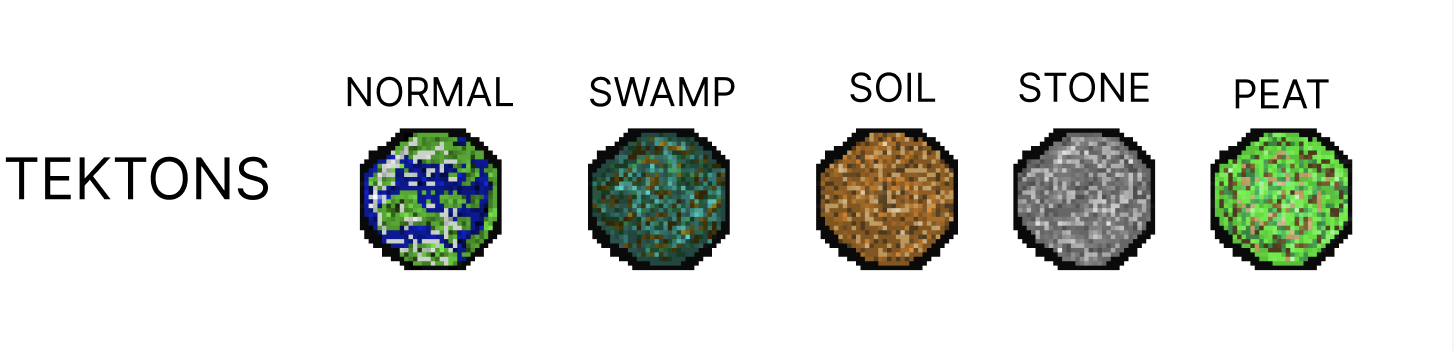
## **11.1** **A grafikus interfész**

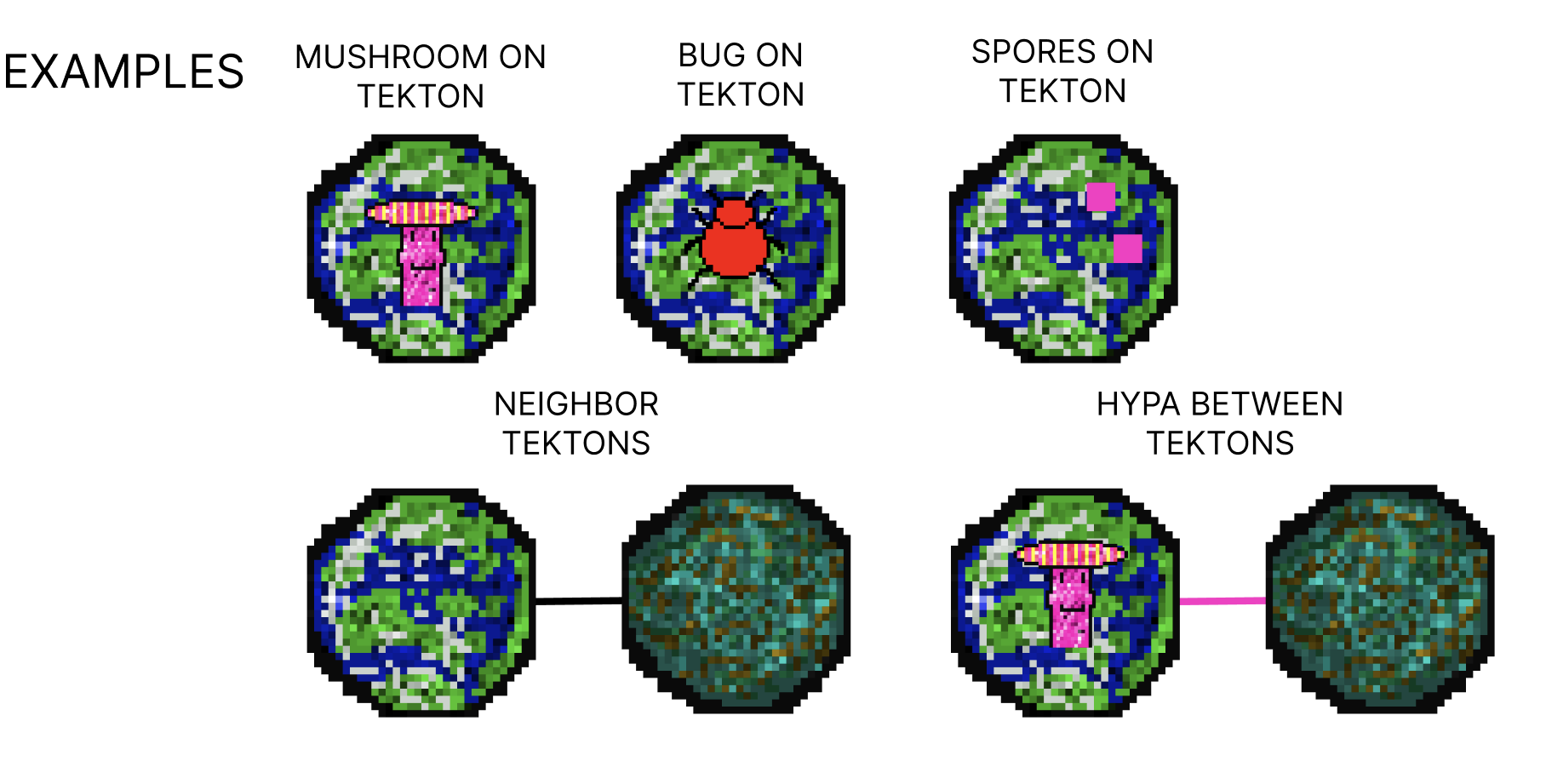
## A Bogarak 4 különböző színűek lehetnek, mindegyik szín egy különböző játékost jelöl. A játékosok max száma 4. A Bogarak textúrája függ hogy milyen spóra hatás alatt áll az adott bogár.

## 

A Gombák a fajtájuktól és koruktól függően különböző textúrájuak. A Spórájuk amit dobnak egy egyszerű primitív kör. A Fonaluk amit növesztenek egy egyszerű vonal.

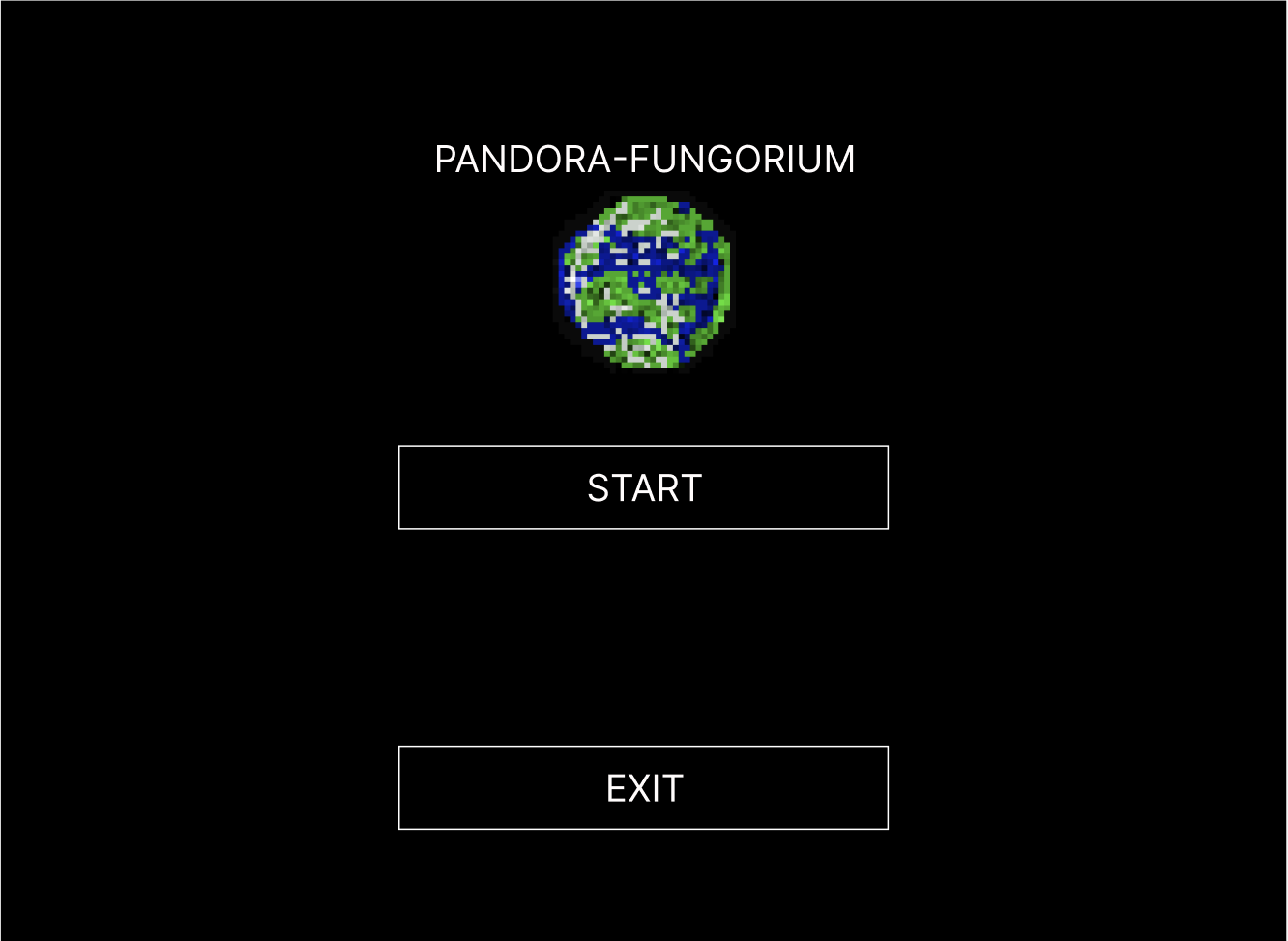






STARTMENU

A játék indításakor megjelenő képernyő. Két gomb van rajta egy cím és egy logó. A Start gomb a játék beállítások menübe vezet, míg az Exit gomb megnyomására kilép a játékból.



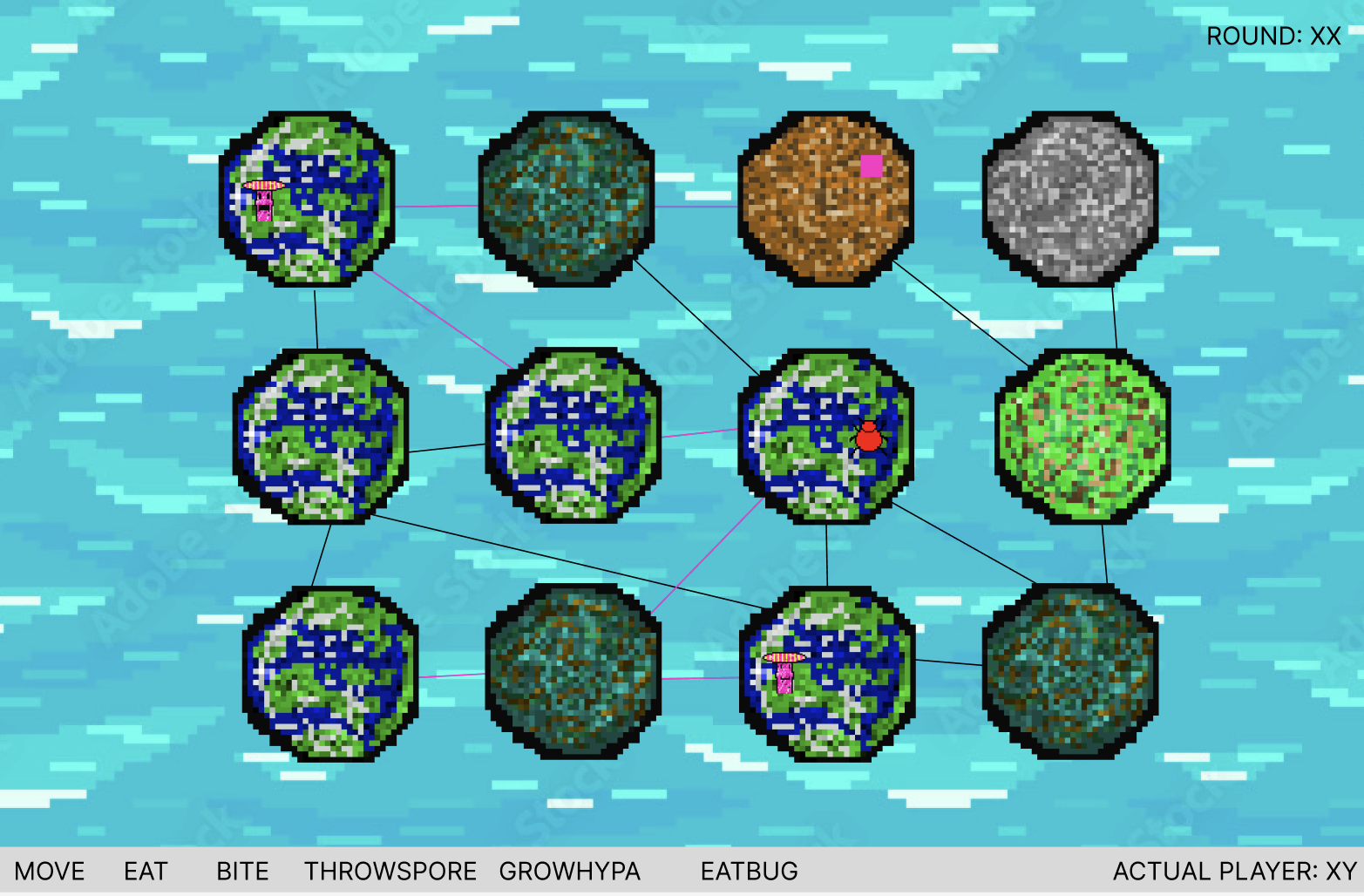
GAMESETTINGS MENU:

A játék beállítása menüben lehet megadni a gombászok és rovarászokat. A gombász felirat alatti “+” gomb megnyomásával új gombászt ad hozzá. Ha a gombász típusának a gombjára kattint a felhasználó változik a gombász gombájának a típusa. A gombász nevét a NAME label alatti textboxba lehet megadni. Rovarászt ugyanígy lehet hozzáadni. A rovarász színét a rovarász szín gombjára kattintva lehet változtatni. A rovarász nevét a NAME label alatti textboxba lehet megadni. A start gombra a játék elindulni, míg az exit gombbal vissza lehet lépni a főmenübe.



Játék Futása:

A játék közben így fog kinézni a teljes pálya. A jobb felső sarokban a körszámláló, jobb alsó sarokban az aktuális játékos neve íródik ki. Alul egy sávban gombok vannak. Mindegyik gomb egy akció. Az aktuális játékos először mindig választ egy akciót, a program ellenőrzi hogy olyan akciót választott a játékos amit végre tud hajtani. Ezután a játékosnak az akcióhoz adott inputot kell megadjon a pályára kattintással. // Pl. Aktuális játékos bogarász, választott akció EAT, Ki kell választani egy bogarat, hogy melyikkel egyen, majd a spórát, amit meg akar enni.//



## **11.2** **A grafikus rendszer architektúrája**

### **11.2.1** **A felület működési elve**

A felület működési elve **pull** alapú. Minden felhasználói akció után a View információt kér a modelltől, és újra rajzolja az egész játékteret.

Minden kattintható modellbeli objektumnak van egy HitBox-a, amely az objektum példányosításakor observer szerűen feliratkozik a modellbeli elemre, értesül annak változásairól (pl. Bug mozog, állapot változik, stb..), ezeknek a HitBox-oknak a felelőssége

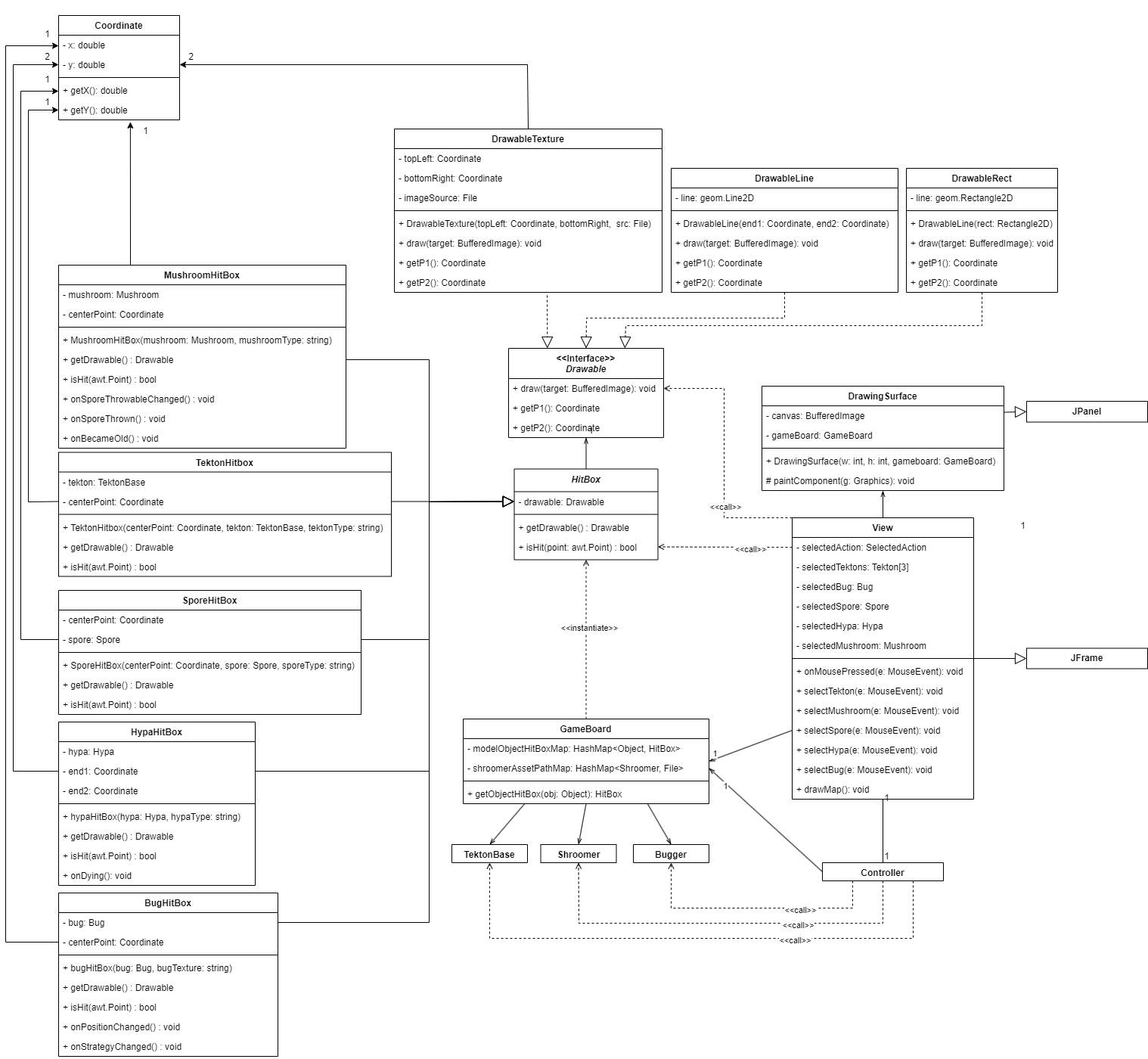
* a grafikai megjelenítéshez szükséges Drawable példány tárolása, frissítése az objektum állapotának megfelelően
* A saját pozíció és esetleg egyéb geometriai jellemzők frissítése a modellbeli elem eseményeire reagálva
* annak eldöntése, hogy az (x, y) koordinátájú kattintás rá esett-e

A játék során a soron következő játékos gombokkal kiválaszthatja az elvégezni kívánt akciót, pl. ha Bugger játékos következik, kiválaszthatja a move, eat, bite vagy skip opciókat. Miután kiválasztott egy akciót, a paraméterek sorrendjét figyelembe véve kell kattintással kiválasztani a szükséges objektumokat (a View ehhez az objektumok hitboxát használja). Ha minden paraméter a helyén van, a View szól a Controllernek, ami elindítja a modellbeli akciót. Ha az akció sikeres, a View újra rajzolja a pályát (ehhez a HitBoxoktól elkéri a Drawable példányukat, és a draw() metódusukat hívja. A HitBoxok a kirajzolásig már értesülnek a modellbeli változásokról eseményeken keresztül, és frissítik a Drawable példányukat), ha nem, jelzi az akció sikertelenségét a felhasználónak.

Példa a paraméterek kiválasztására:

Bugger játékos következik, gombbal kiválasztja, hogy move akciót szeretne végrehajtani. Először egy Bug példányt kell kiválasztani: kattintásra a GameBoard tektonjain végigmegyünk, lekérdezzük a Bug-jaikat, és az ezekhez GameBoard-ban tárolt hitboxtól megkérdezzük, hogy az (x,y) koordinátájú kattintás rá esett e. Ha igen, a felhasználó kiválasztotta a Bugot. (a View nem vizsgálja, hogy ez a Bug a soron következő Buggeré-e, ez a modellben úgyis kiderül, és az akció sikertelen lesz, ha nem saját Bugot választott ki.) Ha sikeres volt a Bug kiválasztása, egy Tekton példányt kell kiválasztani. Kattintásra GameBoard Tektonjain végigmegyünk, és az és az ezekhez GameBoard-ban tárolt hitboxtól megkérdezzük, hogy az (x,y) koordinátájú kattintás rá esett e. Ha igen, a Tekton kiválasztása sikeres volt (a View nem vizsgálja, hogy erre a Tektonra a soron következő Bugger léphet-e, ez a modellben úgyis kiderül, és az akció sikertelen lesz, ha nem).

### **11.2.2** **A felület osztály-struktúrája**



## **11.3** **A grafikus objektumok felsorolása**

### **11.3.1** **<<Interface>> *Drawable***

#### **Felelősség**

Interfész. A grafikus objektumok rajzolásához szükséges metódusok definiálása.

#### **Metódusok**

* + **+ draw(target:BufferedImage): void:** Kirajzolja az objektumot a cél BufferedImage-re.
  + **+ getP1(): Coordinate:** Visszaadja az objektum első koordinátáját.
  + **+ getP2(): Coordinate:** Visszaadja az objektum második koordinátáját

### **11.3.2** **DrawableLine**

#### **Felelősség**

Vonalak grafikus megjelenítése.

#### **Interfészek**

Drawable

#### **Attribútumok**

* + **- line: Line2D.Double:** A vonal geometriai reprezentációja.
  + **- p1: Coordinate:** A vonal kezdőpontja.
  + **- p2: Coordinate:** A vonal végpontja.

#### **Metódusok**

* + **+DrawableLine(end1: Coordinate, end2: Coordinate ):** Konstruktor, létrehozza a vonalat a megadott koordináták alapján.
  + **+ draw(target: BufferedImage): void:** Implementáció.
  + **+ getP1(): Coordinate** Visszaadja a vonal kezdőpontját
  + **+ getP2():** **Coordinate:** Visszaadja a vonal végpontját.

### **11.3.3** **DrawableRect**

#### **Felelősség**

Téglalapok grafikus megjelenítése.

#### **Interfészek**

Drawable

#### **Attribútumok**

* + **- rect: Rectangle2D.Double:** A téglalap geometriai reprezentációja.
  + **- p1: Coordinate:** A téglalap bal felső sarka.
  + **- p2: Coordinate:** A téglalap jobb alsó sarka.

#### **Metódusok**

* + **+DrawableRect(rect: Rectangle2D):** Létrehoz egy új DrawableRect objektumot egy meglévő Rectangle2D objektumból.
  + **+ draw(target: BufferedImage): void:** Kirajzolja a téglalapot a cél BufferedImage-re fekete színnel.
  + **+ getP1(): Coordinate:** Visszaadja a téglalap bal felső sarkát
  + **+ getP2(): Coordinate:** Visszaadja a téglalap jobb alsó sarkát

### **11.3.4** **DrawableTexture**

#### **Felelősség**

Textúrák grafikus megjelenítése.

#### **Interfészek**

Drawable

#### **Attribútumok**

* + **- topLeft: Coordinate:** A textúra bal felső sarka.
  + **- bottomRight: Coordinate:** A textúra jobb alsó sarka.
  + **- image: BufferedImage:** A megjelenítendő kép.

#### **Metódusok**

* + **+DrawableTexture(topLeft: Coordinate, bottomRight: Coordinate , image: BufferedImage )**
  + **+DrawableTexture(topLeft: Coordinate, bottomRight: Coordinate , imageFile: File)**
  + **+void draw(target: BufferedImage):** Kirajzolja a textúrát a megadott pozícióba és méretben a cél BufferedImage-re.
  + **+ getP1(): Coordinate:** Visszaadja a textúra bal felső sarkának koordinátáit.
  + **+ getP2(): Coordinate:** Visszaadja a textúra jobb alsó sarkának koordinátáit.

### **11.3.5** ***Hitbox***

#### **Felelősség**

Különböző típusú objektumoknak a hitboxait kezeit.

#### **Attribútumok**

* + **-drawable: Drawable -** A hitboxhoz tartozó grafikus elem.

#### **Metódusok**

* + **+ getDrawable(): Drawable:** Visszaadja a hitboxhoz tartozó grafikus objektumot.
  + **+ isHit(point: awt.Point): boolean:** Megmondja, hogy a megadott pont (x,y) benne van-e a hitboxban.

### **11.3.6** **Coordinate**

#### **Felelősség**

Koordináták tárolása és kezelése.

#### **Attribútumok**

* + **- x: double:** A pont X koordinátája.
  + **- y: double:** A pont Y koordinátája.

#### **Metódusok**

* + **+Coordinate(x: double, y: double):** Létrehoz egy új koordináta objektumot a megadott x és y értékekkel.
  + **+ getX(): double:** Visszaadja a pont X koordinátáját.
  + **+ getY(): double:** Visszaadja a pont Y koordinátáját.

### **11.3.7** **MushroomHitBox**

#### **Felelősség**

A modellbeli Mushroom típusú objektumokhoz tartozó hitbox. Feladata a Mushroom-ok interaktív terének kezelése. Ezenkívűl az objektumok állapotának figyelése és vizuális megjelenítése.

#### **Ősosztályok**

HitBox

#### **Attribútumok**

* + **- mushroom: Mushroom** :a hozzárendelt Mushroom példány.
  + **- centerPoint: Coordinate** :A hitbox középpontja.

#### **Metódusok**

* + **+ getDrawable(): Drawable** :Visszaadja a kirajzolható objektumot.
  + **isHit(point: awt.Point): boolean:** Visszaadja egy boolean formájában, hogy a paraméterül kapott képpont benne van-e a hitbox-ban.
  + **+ onSporeThrowableChanged(): void** :Akkor hívódik, ha változott a Mushroom spóradobási állapota, és ezért újra kell rajzolni.
  + **+ onSporeThrown(): void** :Akkor hívódik, ha spórát szórt a Mushroom, és ezért újra kell rajzolni.
  + **+ onBecameOld(): void** : Akkor hívódik meg, ha megöregedett a Mushroom, és ezért újra kell rajzolni.

### **11.3.8** **TektonHitbox**

#### **Felelősség**

A modellbeli TektonBase típusú objektumokhoz tartozó hitbox. Feladata a Tekton-ok interaktív terének kezelése és vizuális megjelenítése.

#### **Ősosztályok**

HitBox

#### **Attribútumok**

* + **- tekton: TektonBase** :a hozzárendelt TektonBase objektum.
  + **- centerPoint: Coordinate** :A hitbox középpontja.

#### **Metódusok**

* + **+ getDrawable(): Drawable** :Visszaadja a kirajzolható objektumot.
  + **+ isHit(awt.Point): boolean**:Visszaadja egy boolean formájában, hogy a paraméterül kapott képpont benne van-e a hitbox-ban.

### **11.3.9** **SporeHitBox**

#### **Felelősség**

A modellbeli Spore típusú objektumokhoz tartozó hitbox. Feladata a Spore-ok interaktív terének kezelése és vizuális megjelenítése.

#### **Ősosztályok**

HitBox

#### **Attribútumok**

* + **- spore: Spore** :a hozzárendelt Spore példány.
  + **- centerPoint: Coordinate**:A hitbox középpontja.

#### **Metódusok**

* + **+ getDrawable(): Drawable**:Visszaadja a kirajzolható objektumot.
  + **+ isHit(awt.Point): boolean**:Visszaadja egy boolean formájában, hogy a paraméterül kapott képpont benne van-e a hitbox-ban.

### **11.3.10** **HypaHitBox**

#### **Felelősség**

A modellbeli Hypa típusú objektumokhoz tartozó hitbox. Feladata a Hypa-k interaktív terének kezelése. Ezenkívűl az objektumok állapotának figyelése és vizuális megjelenítése.

#### **Ősosztályok**

HitBox

#### **Attribútumok**

* + **- hypa: Hypa**:a hozzárendelt Hypa objektum.
  + **- end1: Coordinate:** A hitbox egyik végpontja.
  + **- end2: Coordinate:** A hitbox másik végpontja.

#### **Metódusok**

* + **+ getDrawable(): Drawable**:Visszaadja a kirajzolható objektumot.
  + **+ isHit(awt.Point): boolean**:Visszaadja egy boolean formájában, hogy a paraméterül kapott képpont benne van-e a hitbox-ban.
  + **+ onDying(): void** :Akkor hívódik, amikor a Hypa objektum elpusztul, és ezért újrarajzolásra van szükség.

### **11.3.11** **BugHitBox**

#### **Felelősség**

A modellbeli Bug típusú objektumokhoz tartozó hitbox. Feladata a Bug-ok interaktív terének kezelése. Ezenkívűl az objektumok állapotának figyelése és vizuális megjelenítése.

#### **Ősosztályok**

HitBox

#### **Attribútumok**

* + **- bug: Bug**:a hozzárendelt Mushroom példány.
  + **- centerPoint: Coordinate**:A hitbox középpontja.

#### **Metódusok**

* + **+ getDrawable(): Drawable**:Visszaadja a kirajzolható objektumot.
  + **+ isHit(awt.Point): boolean**:Visszaadja egy boolean formájában, hogy a paraméterül kapott képpont benne van-e a hitbox-ban.
  + **+ onPositionChanged(): void**:Akkor hívódik, ha változott a Bug pozíciója, és ezért újra kell rajzolni.
  + **+ onStrategyChanged(): void**:Akkor hívódik, ha a Bug-ot újra kell rajzolni, mert megevett egy spórát és ezáltal változott a stratégiája.

### **11.3.12** **Gameboard**

#### **Felelősség**

A játékban résztvevő játékosok, tektonok nyilvántartása, a modell összefogása oly módon, hogy a Controller tudja vezérelni és a View meg tudja jeleníteni az aktuális játékállást.

#### **Attribútumok**

* + - **modelObjectHitboxMap: Map<Object, HitBox>:** a játékban szereplő összes objektumra referencia párosítva a hozzá tartozó hitbox-szal.Kulcs az objektum, érték a hozzá tartozó Hitbox.
  + **- shroomerAssetPathMap: Map<Shroomer, File>:** A Shroomer típusú karakterekhez tartozó grafikus asset-ek (képek) elérési útvonalait tárolja. Kulcs a Shroomer objektum, érték a hozzá tartozó mappa helye.
* **Metódusok**
  + + **getObjectHitBox(Object obj): HitBox:** Visszaadja a paraméterként kapott objektumhoz tartozó hitboxot. Ha az objektum nincs regisztrálva a rendszerben, null értékkel tér vissza.

### **11.3.13** **View**

#### **Felelősség**

A felhasználói felület vizuális rétegét biztosítja, kezeli a kattintásokat és kirajzolja a pályát.

#### **Attribútumok**

* + **- drawingSurface: DrawingSurface:** a kirajzolt pálya képe egy JPanel egységbe csomagolva
  + **- selectedAction: SelectedAction** : Az aktuálisan kiválasztott művelet.
  + **- selectedTektons: Tekton[3]**: Kiválasztott Tekton objektumok tömbje.
  + **- selectedBug: Bug**: A kiválasztott Bug objektum.
  + **- selectedSpore: Spore**: A kiválasztott Spore objektum.
  + **- selectedHypa: Hypa**: A kiválasztott Hypa objektum.
  + **- selectedMushroom: Mushroom**: A kiválasztott Mushroom objektum.

#### **Metódusok**

* + **+ selectTekton(e: MouseEvent): void**:Akkor hívódik, ha kattintáskor Tekton objektum került kiválasztásra.
  + **+ selectMushroom(e: MouseEvent): void**:Akkor hívódik, ha kattintáskor Mushroom objektum került kiválasztásra.
  + **+ selectSpore(e: MouseEvent): void**:Akkor hívódik, ha kattintáskor Spore objektum került kiválasztásra.
  + **+ selectHypa(e: MouseEvent): void**:Akkor hívódik, ha kattintáskor Hypa objektum került kiválasztásra.
  + **+ selectBug(e: MouseEvent): void**:Akkor hívódik, ha kattintáskor Bug objektum került kiválasztásra.

### **11.3.14** **DrawingSurface**

#### **Felelősség**

A canvas BufferedImage-nek biztosít egy JPanel leszármazott wrappert, hogy köré lehessen tenni egy JScrollPane-t, ezáltal az egér eseményeket csak a játéktér kontextusában, személyre szabva tudjuk kezelni.

#### **Ősosztályok**

JPanel

#### **Attribútumok**

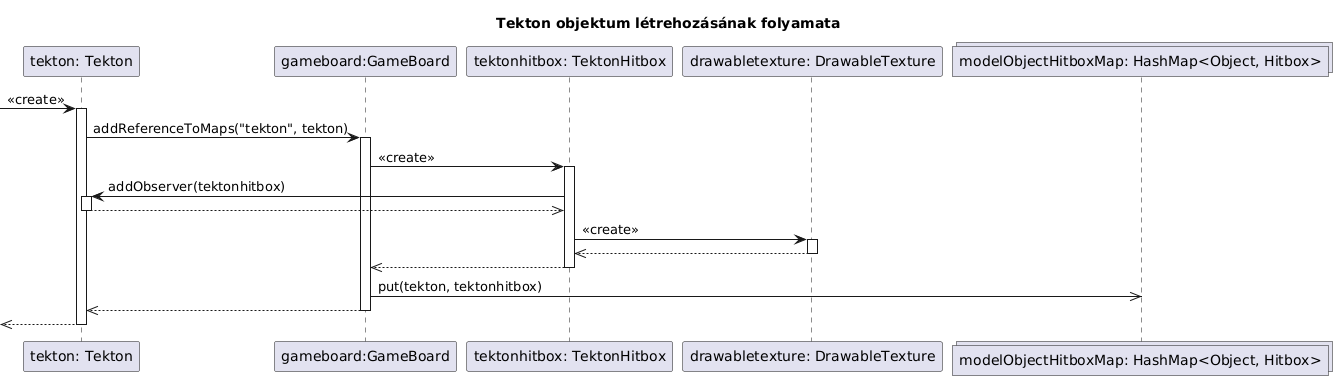
* + **- canvas: BufferedImage**:a rajzfelület
  + **- gameBoard: GameBoard**:referencia a játéktáblára, amitől le tudja kérdezni a játék elemeinek hitboxát a rajzoláshoz..

#### **Metódusok**

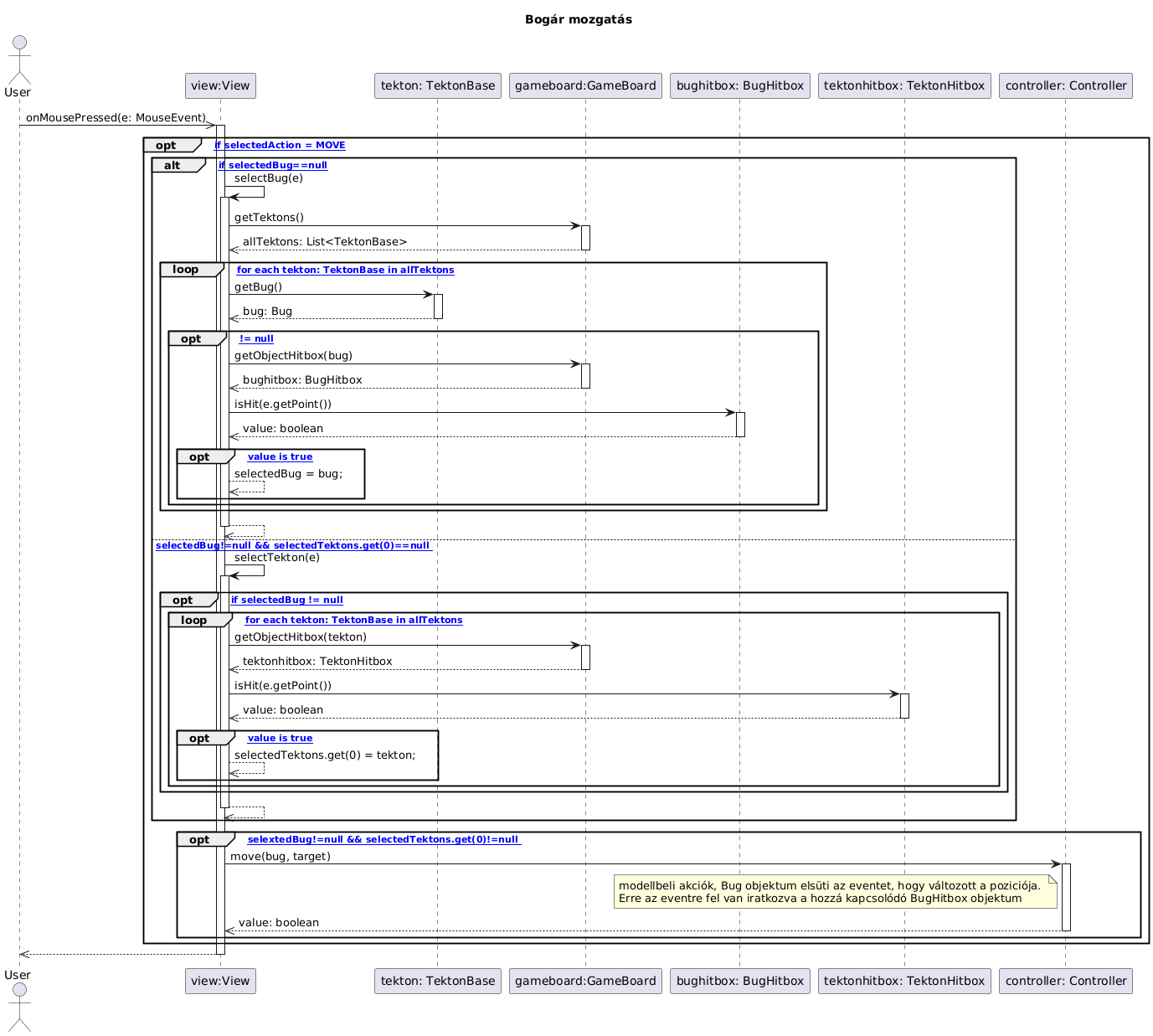
* + **+ DrawingSurface(w: int, h: int, gameboard: GameBoard):** Konstruktor, létrehozza a canvas-t és beállítja a saját GameBoard referenciáját a paraméterként kapottra.
  + **# paintComponent(g: Graphics): void**: Erre a metódusra támaszkodik minden alkalommal az alkalmazás, ha a panelt újra kell rajzolni (explicit repaint() hívásra vagy ablak átméretezése/mozgatása esetén). Ebben a metódusban a DrawingSurface a gameboard-tól elkéri a játékban szereplő objektumok hitboxait, majd azoktól a drawable-jeiket, és a canvas-t paraméterként adva a draw(BufferedImage) függvényüknek kirajzoltatja őket a canvas-re.

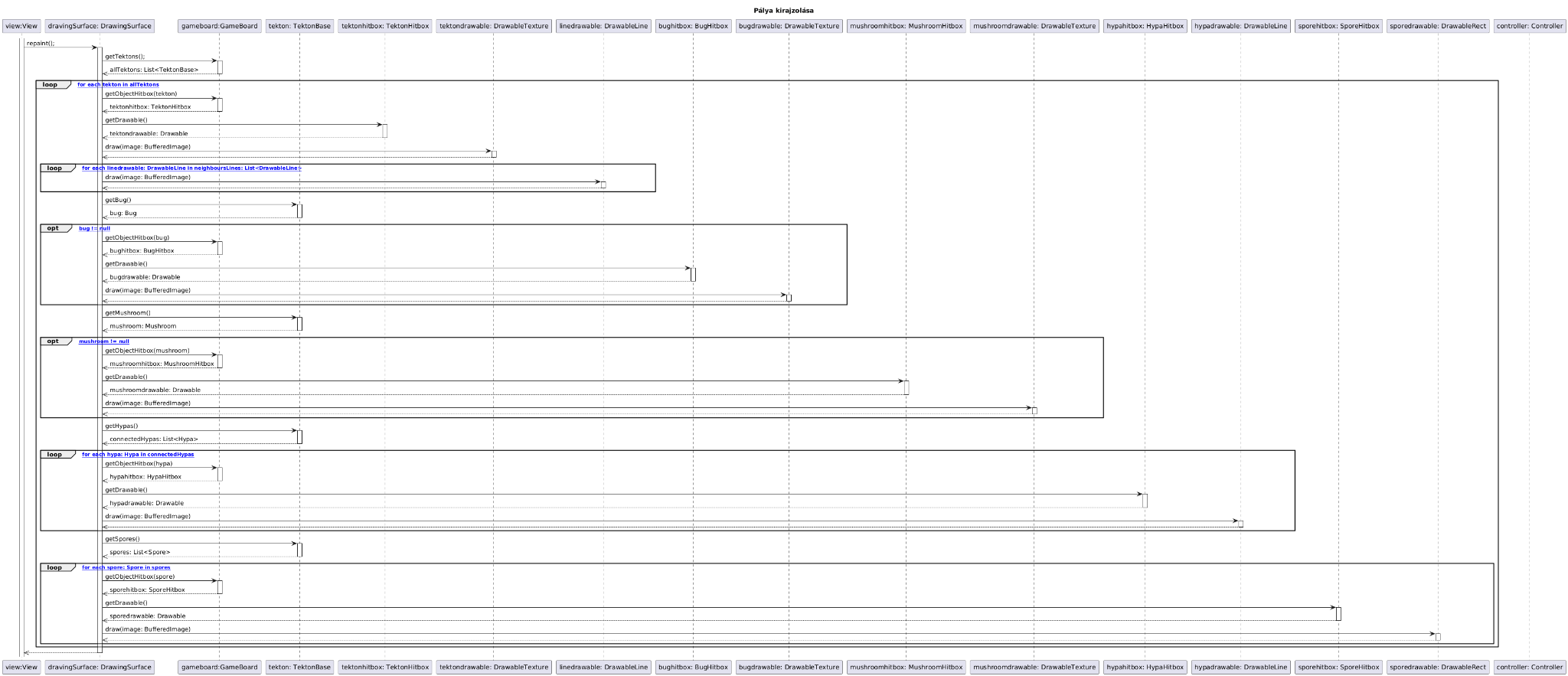
## **11.4** **Kapcsolat az alkalmazói rendszerrel**

Szekvencia diagram arról, hogy hogyan kerül beállításra a kapcsolat a modellbeli objektumok és a grafikus elemek között:

**

Szekvencia diagram arról, hogy hogyan zajlik egy akció végrehajtása a megjelenítés oldaláról. Egy kiválasztott akció után megfelelő objektumokra (hitboxokra) való kattintásokra vár a view, majd azokat átadva szól a controllernek.

**

Szekvencia diagram, amely bemutatja az egész pálya kirajzolásának folyamatát.

## Az ablak felépítésének szekvenciadiagramja:

## **11.5** **Napló**

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2025. 04. 29. 12:30 | 1,5 óra | Kőműves  Szabó | Grafikus megjelenítés osztályainak tervezése |
| 2025. 04. 29. 10:00 | 2 óra | Fekete  Kőműves  Katona  Lipóczi | Konzultáció |
| 2025. 04. 30. 13:00 | 1 óra | Szabó  Katona  Kőműves | Grafikus megjelenítés tervezése |
| 2025. 04. 30. 22:00 | 3 óra | Lipóczi | Drawables, Hitbox és Coordinate osztályok implementálása |
| 2025. 05. 01. 20:00 | 2 óra | Lipóczi | Osztályok leírásai |
| 2025. 05. 02. 20:00 | 3 óra | Fekete | Osztályleírások |
| 2025. 05. 02. 20:00 | 2 óra | Lipóczi | Osztályok leírásának javítása |
| 2025.05.03 12:00 | 10 óra | Katona | PixelArtok megtervezése és rajzolása, menü rendszerek tervezése, játék scene-kre példa készítés. |
| 2025.05.04 17:30 | 1,5 óra | Szabó Kőműves | Grafikus felület felépítésének tervezése |