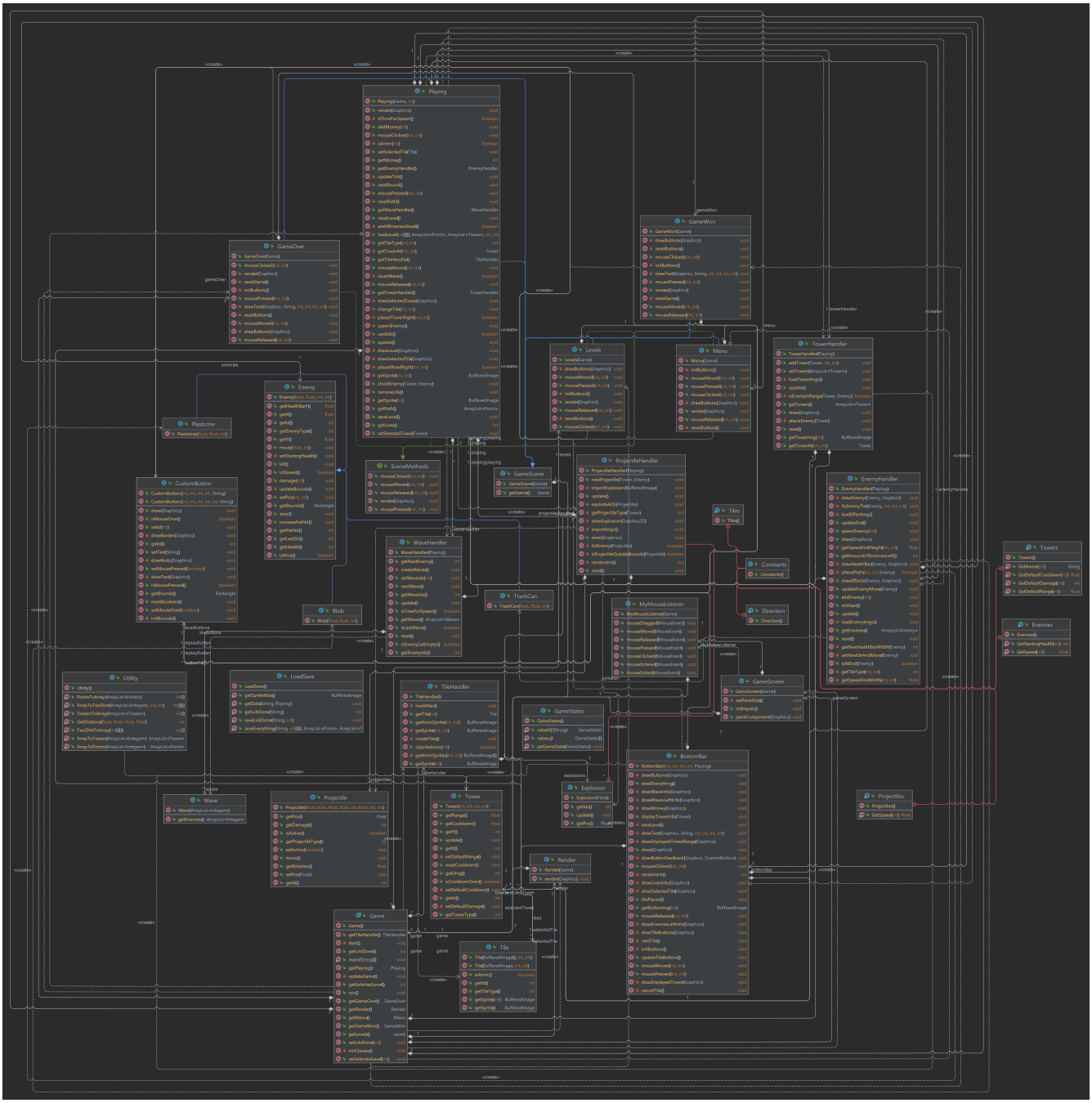
Programozás alapjai 3

NHZ

Tower Defense

Vass Anatol Botond
2022. November 27.

1. Osztálydiagram:



2. Osztályok, és metódusaik leírása:

1. Enemy:

1. Röviden:

- A pályán haladó szörnyeknek az absztrakt ősosztálya.

2. Változók:

- protected float x, y;
- protected Rectangle bounds;
- protected int health;
- protected int maxHealth;
- protected int id;
- protected int pathId = 0;
- protected int enemyType;
- protected int lastDir;
- protected boolean alive = true;
- protected int slowTickMax = 2 * 60;
- protected int slowTick = slowTickMax;

3. Metódusok:

- public Enemy(float x, float y, int id, int enemyType)
  - Konstruktor
- private void setStartingHealth()
  - Beállítja a szörny életét a típusa alapján
- public void damage(int dmg)
  - Levon a szörny életéből a paraméterként megadott mennyiséget, ha ezután a szörny élete nulla alá csökken akkor „megöli”.
- public void slow()
  - A szörny slowTick változóját nullára állítja.
- public void move(float speed, int dir)
  - A szörny mozgatását valósítja meg a koordinátái változtatásával a haladási irányt is figyelembe véve. Ha a szörny slowTick változója kisebb mint annak a maximuma akkor lassabban halad.
- public void kill()
  - „Megöli” a szörnyet a megfelelő boolean változó false-ra állításával és az élete nullázásával. Ezt a metódust csak akkor hívjuk, ha a szörny elérte az út végét.
- private void updateBounds()
  - A szörny „Hitbox”-át frissíti a pozíciójához.
- public void setPos(int x, int y)
  - A pozícióját állítja át a paraméterként kapott értékekre, csak a szörny pozíciójának fixálásához kell, hogy ne lépje túl a maximum koordinátát.
- public float getHealthBarF()

        - Viszaad egy float értéket, melyet egy másik metódusban a szörny életének kirajzolásához használunk.
    - public float getX()
        - Visszaadja a szörny x koordinátáját.
    - public float getY()
        - Visszaadja a szörny y koordinátáját.
    - public Rectangle getBounds()
        - Visszaadja a szörny hitbox-át.
    - public int getHealth()
        - Visszaadja a szörny jelenlegi életmennyiségét.
    - public int getId()
        - Visszaadja a szörny azonosítóját.
    - public int getEnemyType()
        - Visszaadja a szörny típusát.
    - public int getLastDir()
        - Visszaadja a szörny legutóbbi haladási irányát.
    - public int getPathId()
        - Visszaadja azon út elemnek az azonosítóját amelyiken áll.
    - public boolean isAlive()
        - Visszaadja, hogy a szörny él-e.
    - public boolean isSlowed()
        - Visszaadja, hogy a szörny le van-e lassítva.
    - public void increasePathId()
        - Növeli az út azonosítót egyel.

2. Blob:

1. Leírás:

- Az Enemy osztály leszármazottja, a BLOB típusú szörny.

2. Változók:

- -

3. Metódusok:

- public Blob(float x, float y, int id)
    - Konstruktor, az ősosztály konstruktorát hívja meg a megadott paraméterekkel és a szörny saját típusával.

3. Plasticine:

1. Leírás:

- Az Enemy osztály leszármazottja, a PLASTICINE típusú szörny.

2. Változók:

- -

3. Metódusok:

- public Plasticine(float x, float y, int id)
  - Konstruktor, az ősosztály konstruktorát hívja meg a megadott paraméterekkel és a szörny saját típusával.

4. TrashCan:

1. Leírás:

- Az Enemy osztály leszármazottja, a TRASHCAN típusú szörny.

2. Változók:

- -

3. Metódusok:

- public TrashCan(float x, float y, int id)
  - Konstruktor, az ősosztály konstruktorát hívja meg a megadott paraméterekkel és a szörny saját típusával.

5. Wave:

1. Leírás:

- Szörnyek azonosítójának sorozatát tároló osztály.

2. Változók:

- private ArrayList<Integer> enemies;

3. Metódusok:

- public Wave(ArrayList<Integer> enemies)
  - Konstruktor.
- public ArrayList<Integer> getEnemies()
  - Visszaadja a szörnyek azonosítóját tartalmazó tömböt.

6. Tile:

1. Leírás:

- A pályán látható pályaelemek osztálya. A változó alapján derül ki típusa. Ilyenek például a fű, az út elem és a víz.

2. Változók:

- private BufferedImage[] sprite;
- private int id, tileType;

3. Metódusok:

- public Tile(BufferedImage sprite, int id, int tileType)
  - Konstruktor a nem animált elemeknek.
- public Tile(BufferedImage[] sprite, int id, int tileType)
  - Konstruktor az animált elemeknek
- public BufferedImage getSprite(int animIdx)
  - Visszaadja a paraméterként adott indexű képet. Csak animált elem esetén használatos.
- public BufferedImage getSprite()
  - Visszaadja az elemhez tartozó képet. Csak nem animált elemekhez használatos.
- public boolean isAnim()

        - Visszaadja, hogy az elem animált-e.
    - public int getTileType()
        - Visszaadja az elem típusát.
    - public int getId()
        - Visszaadja az elem azonosítóját.

7. Tower:

1. Leírás:

- A játékban lévő tornyok osztálya.

2. Változók:

- private int x, y, id, towerType, cdwnTick, dmg;
- private float range, cooldown;

3. Metódusok:

- public Tower(int x, int y, int id, int towerType)
    - Konstruktor.
- public void update()
    - Növeli a cooldown számlálót.
- public boolean isCooldownOver()
    - Megnézi, hogy lejárt-e a cooldown.
- public void resetCooldown()
    - Visszaállítja a cooldown-t 0-ra.
- private void setDefaultCooldown()
    - Beállítja az alap cooldown időd a torony típusa alapján.
- private void setDefaultRange()
    - Beállítja az alap hatótávot a torony típusa alapján.
- private void setDefaultDamage()
    - Beállítja az alap sebzést a torony típusa alapján.
- public int getX()
    - Visszaadja a torony x koordinátáját.
- public int getY()
    - Visszaadja a torony y koordinátáját.
- public int getId()
    - Visszaadja a torony azonosítóját.
- public int getTowerType()
    - Visszaadja a torony típusát.
- public int getDmg()
    - Visszaadja a torony sebzését.
- public float getRange()
    - Visszaadja a torony hatótávolságát.
- public float getCooldown()

- Visszaadja a torony alap cooldown értékét.

8. Projectile:

1. Leírás:

- A tornyok lövedékének osztálya.

2. Változók:

- private Point2D.Float pos;
- private int id, projectileType, damage;
- private float xSpeed, ySpeed, rotation;
- private boolean active = true;

3. Metódusok:

- public Projectile(float x, float y, float xSpeed, float ySpeed, int damage, float rotation, int id, int projektileType)
  - Konstruktor
- public void move()
  - Mozgatja a lövedéket.
- public Point2D.Float getPos()
  - Visszaadja a lövedék pozícióját.
- public void setPos(Point2D.Float pos)
  - Beállítja a lövedék pozícióját.
- public int getId()
  - Visszaadja a lövedék azonosítóját.
- public int getDamage()
  - Visszaadja a lövedék sebzését.
- public float getRotation()
  - Visszaadja a lövedék fordulásához szükséges értéket.
- public int getProjectileType()
  - Visszaadja a lövedék típusát.
- public boolean isActive()
  - Visszaadja, hogy a lövedék aktív-e.
- public void setActive(boolean active)
  - Beállítja a lövedék aktivitását a megadott értékre.

9. EnemyHandler:

1. Leírás:

- A szörnyek irányítását végző osztály.

2. Változók:

- private Playing playing;
- private BufferedImage[] enemyImgs;
- private BufferedImage slowEffect;
- private ArrayList<Enemy> enemies = new ArrayList<>();

- private ArrayList<Point> path = new ArrayList<>();
- private int startX, startY, endX, endY;
- private int HealthBarWidth = 28;

3. **Metódusok:**
   - public EnemyHandler(Playing playing)
     - Konstruktor.
   - private void loadEffectImg()
     - Betölti a lassítás effekt képét.
   - private void loadEnemyImgs()
     - Betölti a szörnyek képét.
   - public void initStart()
     - Beállítja a szörnyek indulási pontját.
   - public void updateEnd()
     - Beállítja a szörnyek haladásának végpontját. Ha lerakunk egy utat akkor is ezt hívjuk meg, hogy frissítse a változókat.
   - public void update()
     - Végigmegy a tárolt szörnyeken és amelyik életben van azt mozgatja.
   - public void spawnEnemy(int nextEnemy)
     - Hozzáadja a szörnyek listájához a paraméterként kapott szörnyet.
   - public void updateEnemyMove(Enemy e)
     - Ha még nem indult el a paraméterként kapott szörny akkor elindítja. Ellenőrzi, hogy jó helyen halad-e, ha éppen rossz helyre lépne akkor meghívja az irányváltoztató metódust. Ha a pálya végére ér akkor „megöli” és levon egy életet a játékostól.
   - private void setNewDirAndMove(Enemy e)
     - A haladási irányt állítja át a megfelelőre, majd mozgatja a paraméterként kapott szörnyet.
   - private boolean isNextPath(int x, int y, Enemy e)
     - Megnézi, hogy a következő elem amelyre lépni akar a szörny az megegyezik-e a következő úttal. Ha már a 2-vel utánival egyezik meg, akkor növeli az út elem azonosítóját. Ha egyik sem akkor hamis értékkel tér vissza, egyébként a másik két esetben pedig igazzal.
   - private void fixEnemyTile(Enemy e, int dir, int xCord, int yCord)
     - Arra használjuk, hogy a szörny teljesen belemozogjon az adott négyzetbe, így tudja rendesen ellenőrizni az irányváltást. Azért ellenőrizzük csak jobbra és le, mert mindig a szörny bal felső koordinátáját hasonlítjuk, és ez csak ezekben a haladási irányokban okoz gondot.
   - public boolean isAtEnd(Enemy e)
     - Visszaadja, hogy a paraméterként kapott szörny a pálya végére ért-e.
   - private int getTileType(int x, int y)
     - Visszaadja a paraméterként kapott koordinátájú elem típusát.
   - private float getSpeedAndWidth(int dir, int enemyType)

- Ezt a metódust is az ellenőrzéseknél használjuk, így figyelembe van véve az is, hogy a szörny bal felső sarkához viszonyítunk.

- private float getSpeedAndHeight(int dir, int enemyType)
  - Ezt a metódust is az ellenőrzéseknél használjuk, így figyelembe van véve az is, hogy a szörny bal felső sarkához viszonyítunk.
- public void addEnemy(int enemyType)
  - Hozzáad egy a paraméterként kapott típusú szörnyet a listához, a kezdő pozícióval.
- public void draw(Graphics g)
  - Kirajzolja az életben lévő szörnyeket.
- private void drawEffects(Enemy e, Graphics g)
  - Kirajzolja a szükséges effektet, ha szükséges.
- private void drawHealthBar(Enemy e, Graphics g)
  - Kirajzolja a paraméterként kapott szörny életét.
- private int getNewHealthBarWidth(Enemy e)
  - Visszaadja a paraméterként kapott szörny életcsíkjának új méretét.
- private void drawEnemy(Enemy e, Graphics g)
  - Kirajzolja a paraméterként kapott szörnyet.
- public ArrayList<Enemy> getEnemies()
  - Visszaadja a tárolt szörnyek listáját.
- public int getAmountOfEnemiesLeft()
  - Visszaadja, hogy még mennyi szörny van a pályán.
- public void reset()
  - Visszaállítja az EnemyHandlert alaphelyzetbe.

10. TowerHandler:

1. **Leírás:**
   - A tornyok irányítását végző osztály.
2. **Változók:**
   - private Playing playing;
   - private BufferedImage[] towerImgs;
   - private ArrayList<Tower> towers = new ArrayList<>();
   - private int towerAmount = 0;
3. **Metódusok:**
   - public TowerHandler(Playing playing)
     - Konstruktor.
   - private void loadTowerImgs()
     - Betölti a tornyok képeit.
   - public BufferedImage getTowerImg(int id)
     - Visszaadja a paraméterként kapott azonosítóhoz tartozó torony képet.
   - public void draw(Graphics g)

        - Kirajzolja a tornyokat.
    - public Tower getTowerAt(int x, int y)
        - Visszaadja a paraméterként kapott koordinátákon álló tornyot, ha nem áll ott egy sem akkor null értékkel tér vissza.
    - public void addTower(Tower selectedTower, int xPos, int yPos)
        - Hozzáadja a paraméterként kapott tornyot a paraméterként kapott pozícióra, és növeli a tárolt tornyok mennyiségét.
    - public void update()
        - Meghívja a tornyok update metódusát és az attackEnemy metódust.
    - private void attackEnemy(Tower t)
        - Ha valamelyik szörny a torony közelében van akkor megtámadja.
    - private boolean isEnemyInRange(Tower t, Enemy e)
        - Visszaadja, hogy a paraméterként kapott szörny a torony hatótávolságában van-e.
    - public ArrayList<Tower> getTowers()
        - Visszaadja a tornyok listáját.
    - public void setTowers(ArrayList<Tower> towers)
        - Beállítja a kapott listát a tornyok listájának.
    - public void reset()
        - Visszaállítja a TowerHandler-t alaphelyzetbe.

11. ProjectileHandler:

1. **Leírás:**
    - A lövedékek irányítását végző osztály.
2. **Változók:**
    - private Playing playing;
    - private ArrayList<Projectile> projectiles = new ArrayList<>();
    - private ArrayList<Explosion> explosions = new ArrayList<>();
    - private BufferedImage[] projImgs, explosionImgs;
    - private int projId = 0;
    - private Random random;
3. **Belső osztály: Explosion**
    1. **Leírás:**
        - A robbanást megvalósító osztály.
    2. **Változók:**
        - private Point2D.Float pos;
        - private int explosionTick = 0, explosionIndex = 0;
    3. **Metódusok:**
        - public Explosion(Point2D.Float pos)
            - Konstruktor
        - public void update()

        - Növeli a számlálót, ha eléri a maximum értéket akkor nullázza és növeli az azonosítót.
    - public Point2D.Float getPos()
        - Visszaadja a robbanás pozícióját.
    - public int getIdx()
        - Visszaadja a robbanás azonsoítóját.

4. **Metódusok:**
    - public ProjectileHandler(Playing playing)
        - Konstruktor
    - private void importImgs()
        - Betölti a lövedékek képeit.
    - private void importExplosion(BufferedImage atlas)
        - Betölti a robbanás effekt képeit.
    - private int randomInt()
        - Egy random int-et ad vissza 100-as határral.
    - public void newProjectile(Tower t, Enemy e)
        - Hozzáad egy új lövedéket a listához, amely a paraméterként kapott toronytól halad a paraméterként kapott szörny felé. Itt állítjuk be a forgatási irányt a nyílvesszőnek is, hogy az a szörny felé nézzen.
    - public void update()
        - Végigmegy az összes aktív lövedéken, ha betalál egy lövedék, akkor inaktiválja, illetve ha bomba akkor meghívja az explodeAOE metódust. Ha a lövedék kiment a pályáról, akkor is inaktiválja a metódus.
    - private void explodeAOE(Projectile p)
        - Ellenőrzi, hogy valamelyik szörny a robbanás hatáskörében van-e, a igen akkor megsebzi.
    - private boolean hitEnemy(Projectile p)
        - Ellenőrzi, hogy a paraméterként kapott lövedék eltalált-e egy szörnyet, ha igen akkor megsebzi. Ha meghalt akkor random esetben pénzt kaphatunk, illetve ha hóval lőtték meg a szörnyet akkor belassul. Igazzal tér vissza ha talált, hamissal ha nem.
    - private boolean isProjectileOutsideBounds(Projectile p)
        - Ellenőrzi, hogy a paraméterként kapott lövedék a pályán kívül van-e.
    - public void draw(Graphics g)
        - Kirajzolja az aktív lövedékeket és meghívja a robbanásokat kirajzoló metódust.
    - private void drawExplosion(Graphics2D g2d)
        - Kirajzolja a robbanáskat.
    - private int getProjectileType(Tower t)
        - Visszaadja az adott toronyhoz tartozó lövedék típusát.
    - public void reset()
        - Visszaállítja a ProjectileHandler-t alaphelyzetbe.

12. TileHandler:

1. **Leírás:**
   - A pályaelemeket kezelő osztály.
2. **Változók:**
   - public Tile WATER,GRASS,ROAD;
   - public BufferedImage atlas;
   - public ArrayList<Tile> tiles = new ArrayList<>();
3. **Metódusok:**
   - public TileHandler()
     - Konstruktor
   - private void createTiles()
     - Hozzáadja a tiles listához a pályaelemeket
   - private void loadAtlas()
     - Betölti a főképet.
   - public Tile getTile(int id)
     - Visszaadja a paraméterként kapott azonosítójú elemet.
   - public BufferedImage getSprite(int id)
     - Visszaadja a paraméterként kapott azonosítójú elemhez tartozó képet.
   - public BufferedImage getAnimSprite(int id, int animIdx)
     - Visszaadja a paraméterként kapott azonosítójú animált elemhez tartozó paraméterként kapott azonosítójú animációképet.
   - private BufferedImage[] getAnimSprites(int x, int y)
     - Visszaadja az animált elemhez tartozó képek tömbjét.
   - private BufferedImage getSprite(int x, int y)
     - Visszaadja a paraméterként megadott helyen levő képrészt.
   - public boolean isSpriteAnim(int id)
     - Visszaadja, hogy a paraméterként kapott azonosítójú elem animált-e.

13.WaveHandler:

1. **Leírás:**
   - A hullámokat kezelő osztály.
2. **Változók:**
   - private Playing playing;
   - private ArrayList<Wave> waves = new ArrayList<>();
   - private int enemySpawnTickMax = 60;
   - private int enemySpawnTick = enemySpawnTickMax;
   - private int enemyIdx, waveIdx = -1;
3. **Metódusok:**
   - public WaveHandler(Playing playing)
     - Konstruktor.
   - public void update()
     - Ha nincs még ideje új szörny lerakásának akkor növeli s spawn számlálót.

- private void createWaves()
  - Feltölti a waves listát szörnyek azonosítójával.
- public void nextWave()
  - Növeli a wave azonosítóját és az enemyIdx-et 0-ra állítja.
- public boolean isTimeForSpawn()
  - Visszaadja, hogy ideje van-e lerakni új szörnyet.
- public boolean isEnemyListEmpty()
  - Visszaadja, hogy a szörnyek listája üres-e.
- public boolean isLastWave()
  - Visszaadja, hogy az utolsó wave van-e.
- public ArrayList<Wave> getWaves()
  - Visszaadja a waves listát.
- public int getWaveIdx()
  - Visszaadja a jelenlegi hullám azonosítóját.
- public int getEnemyIdx()
  - Visszaadja a jelenlegi szörny azonosítóját.
- public int getNextEnemy()
  - Visszaadja a következő szörnyet a listából.
- public void setWaveIdx(int waveData)
  - Beállitja a wave azonosítóját a paraméterként kapottra.
- public void reset()
  - Visszaállítja a WaveHandler-t alaphelyzetbe.

14.Constants:

1. **Leírás:**
   - Különböző osztályokhoz tartozó konstans értékeket tartalmaz.
2. **Változók a belső osztályokban:**
   - **Direction:**
     - public static final int LEFT = 0;
     - public static final int UP = 1;
     - public static final int RIGHT = 2;
     - public static final int DOWN = 3;
   - **Tiles:**
     - public static final int WATER_TILE = 0;
     - public static final int GRASS_TILE = 1;
     - public static final int ROAD_TILE = 2;
   - **Enemies:**
     - public static final int TRASHCAN = 0;
     - public static final int BLOB = 1;
     - public static final int PLASTICINE = 2;

- **Towers:**
  - public static final int ARCHER = 0;
  - public static final int CANNON = 1;
  - public static final int ICE = 2;
- **Projectiles:**
  - public static final int ARROW = 0;
  - public static final int BOMB = 1;
  - public static final int SNOW = 2;

3. **Metódusok a belső osztályokban:**
   - **Enemies:**
     - public static float GetSpeed(int enemyType)
       - Visszaadja a paraméterként kapott típusú szörny sebességét.
     - public static int GetStartingHealth(int enemyType)
       - Visszaadja a paraméterként kapott típusú szörny kezdő életét.
   - **Towers:**
     - public static String GetName(int towerType)
       - Visszaadja a paraméterként kapott típusú torony nevét.
     - public static int GetDefaultDamage(int towerType)
       - Visszaadja a paraméterként kapott típusú torony sebzését.
     - public static float GetDefaultRange(int towerType)
       - Visszaadja a paraméterként kapott típusú torony hatótávolságát.
     - public static float GetDefaultCooldown(int towerType)
       - Visszaadja a paraméterként kapott típusú torony cooldown-ját.
   - **Projectiles:**
     - public static float GetSpeed(int type)
       - Visszaadja a paraméterként kapott típusú lövedékhez tartozó sebességet.

15. Utility:

1. **Leírás:**
   - Kisegítő, számoló, átalakító metódusokat tartalmaz.
2. **Változók:**
   - -
3. **Metódusok:**
   - public static int[][] ArrayToTwoDInt(ArrayList<Integer> list, int ySize, int xSize)
     - A paraméterként kapott Integer listát alakítja át, ugyancsak a paraméterben megadott x és y méretű kétdimenziós tömbbé.
   - public static int[] TwoDIntToArray(int[][] twoArr)
     - A paraméterként kapott két dimenziós tömböt adja vissza egy dimenzióként.
   - public static ArrayList<Point> ArrayToPoints(ArrayList<Integer> list)
     - A paraméterként kapott Integer listát adja vissza Point listaként.

- public static int[] PointsToArray(ArrayList<Point> points)
  - A paraméterként kapott Point listát adja vissza int tömbként.
- public static ArrayList<Tower> ArrayToTowers(ArrayList<Integer> list)
  - A paraméterként kapott Integer listát adja vissza Tower listaként.
- public static int[] TowersToArray(ArrayList<Tower> towers)
  - A paraméterként kapott Tower listát adja vissza int tömbként.
- public static int GetDistance(float x1, float y1, float x2, float y2)
  - A paraméterként kapott két koordináta közötti távolságot adja vissza.

16. LoadSave:

1. **Leírás:**
   - Fájlkezelést megvalósító osztály.
2. **Változók:**
   - -
3. **Metódusok:**
   - public static BufferedImage getSpriteAtlas()
     - Visszaadja a főképet.
   - public static void saveLvlsDone(String name, int lvls)
     - Elmenti az elért szintet.
   - public static void saveEverything(String name, int[][] idArr, ArrayList<Point> pathPoints, ArrayList<Tower> towers, int wave, int money, int lives)
     - Elmenti a paraméterként megadott adatokat a paraméterként megadott nevű fájlba.
   - public static void getData(String name, Playing playing)
     - Betölti a paraméterben kapott nevű fájlból a szükséges adatokat a playing-be.
   - public static int getLvlsDone(String name)
     - Visszaadja az elért szintet.

17. BottomBar:

1. **Leírás:**
   - A játék közben az alsó részt kezelő osztály.
2. **Változók:**
   - private int x, y, width, height;
   - private Playing playing;
   - private CustomButton bMenu, bSave, bNextRound, bNextTile, bCancel;
   - private ArrayList<CustomButton> tileButtons = new ArrayList<>();
   - private Tile selectedTile;
   - private Tower selectedTower;
   - private Tower displayedTower;
   - private Random random;
3. **Metódusok:**
   - public BottomBar(int x, int y, int width, int height, Playing playing)

    - Konstruktor
- private void initButtons()
    - Inicializálja a gombokat.
- public void updateTileButtons()
    - Frissíti a lerakható elemeket mutató gombokat.
- private void drawButtons(Graphics g)
    - Kirajzolja a gombokat.
- private void drawTileButtons(Graphics g)
    - Kirajzolja a lerakható elemeket megjelenítő gombokat.
- private void drawButtonFeedback(Graphics g, CustomButton b)
    - Az elemeket megjelenítő gomboknál rajzolja ki a feedback esemény alapján amit kell. (Fehér körvonal...)
- public BufferedImage getButtonImg(int id)
    - Visszaadja a paraméterként adott azonosítóhoz tartozó képet.
- public void draw(Graphics g)
    - Kirajzolja az alsó rész hátterét és meghívja a többi kirajzoló metódust.
- private void drawLivesInfo(Graphics g)
    - Kirajzolja, hogy mennyi életünk van még.
- private void drawWaveInfo(Graphics g)
    - Kirajzolja, hogy rakhatunk-e vagy sem elemeket a pályára, és ha megy a kör akkor meghívja a szörnyekről információt kirajzoló metódust.
- private void drawWavesLeftInfo(Graphics g)
    - Kirajzolja, hogy hanyadik hullámnál tartunk az üsszesből.
- private void drawEnemiesLeftInfo(Graphics g)
    - Kirajzolja, hogy az adott körben mennyi szörny van még a pályán.
- private void drawDisplayedTower(Graphics g)
    - Kirajzolja a toronyról az információkat, amelyiket éppen kiválasztottuk a pályán.
- private void drawDisplayedTowerRange(Graphics g)
    - Kirajzolja a pályán éppen kiválasztott torony hatótávolságát.
- private void drawMoney(Graphics g)
    - Kirajzolja a pénzünkről az információt.
- public void displayTowerInfo(Tower t)
    - Beállítja displayedTower-nek a paraméterként kapott tornyot.
- private void drawText(Graphics g, String text, int x, int y, int width, int height)
    - Szöveg kiírását valósítja meg a paraméterként megadott kerethez viszonyítva szimmetrikusan.
- private int randomInt()
    - Visszaad egy random int értéket 3 határral.
- private void drawSelectedTile(Graphics g)
    - Kirajzolja a kiválasztott elemet.

- public void mouseClicked(int x, int y)
  - Klikkelés eseteit ellenőrzi, és az alapján hív meg más metódusokat.
- private void cancelTile()
  - A selected dolgokat nullázza.
- public void tilePlaced()
  - A selected dolgokat nullázza és üressé teszi az első gombot
- private void nextTile()
  - Ha van elegendő pénzünk, akkor levon belőle, majd meghívja az elemgombokat frissítő metódust.
- private void saveLevel()
  - Meghívja a playing saveLevel metódusát.
- public void mouseMoved(int x, int y)
  - Az egér helyzetét ellenőrzi,ha valamelyik gomb felett van akkor annak megfelelően rajzol.
- public void mousePressed(int x, int y)
  - Az egér helyzetét ellenőrzi,ha valamelyik gomb felett és lenyomva van az egér akkor annak megfelelően rajzol.
- public void mouseReleased(int x, int y)
  - Reseteli a gombok boolean értékeit.
- public void resetEverything()
  - Visszaállítja a BottomBar-t alaphelyzetbe.

18.CustomButton:

1. **Leírás:**
   - Egyedi gombokat megvalósító osztály
2. **Változók:**
   - public int x, y, width, height, id;
   - private String text;
   - private Rectangle bounds;
   - private boolean mouseOver, mousePressed;
3. **Metódusok:**
   - public CustomButton(int x, int y, int width, int height, String text)
     - Sima gomb konstruktora.
   - public CustomButton(int x, int y, int width, int height, int id, String text)
     - Több gomb egyszeri létrehozására használható konstruktor.
   - private void initBounds()
     - Inicializálja a gomb határait.
   - public void draw(Graphics g)
     - Meghívja a gomb különböző részeit kirajzoló metódusokat.
   - private void drawBorder(Graphics g)
     - Kirajzolja a gomb keretét.

- private void drawBody(Graphics g)
  - Kirajzolja a gomb testét.
- private void drawText(Graphics g)
  - Kirajzolja a gomb szövegét.
- public void resetBooleans()
  - Visszaállítja a gomb boolean értékeit.
- public void setMouseOver(boolean mouseOver)
  - Átállíja a gomb mouseOver értékét a paraméterként kapottra.
- public void setMousePressed(boolean mousePressed)
  - Átállíja a gomb mousePressed értékét a paraméterként kapottra.
- public boolean isMouseOver()
  - Visszaadja a mouseOver értékét.
- public boolean isMousePressed()
  - Visszaadja a mousePressed értékét.
- public Rectangle getBounds()
  - Visszaadja a gomb bounds változójának értékét.
- public int getId()
  - Visszaadja a gomb azonosítóját.
- public void setText(String text)
  - Átállítja a gomb szövegét.
- public void setId(int id)
  - Átállítja a gomb azonosítóját.

19.Game:

1. **Leírás:**
   - A játék futtatását és megjelenítését megvalósító főosztály. Jfram-et örökli és a Runnable-t implementálja.
2. **Változók:**
   - private static final long serialVersionUID = 1L;
   - private GameScreen gameScreen;
   - private Thread gameThread;
   - private final double FPS = 120.0;
   - private final double UPS = 60.0;
   - private Render render;
   - private Menu menu;
   - private ArrayList<Playing> playing = new ArrayList<>();
   - private Levels levels;
   - private GameOver gameOver;
   - private GameWon gameWon;
   - private TileHandler tileHandler;
   - private int selectedLevel = -1;

- private int lvlsDone;

3. **Metódusok:**
   - public Game()
     - Konstruktor.
   - public static void main(String[] args)
     - Main függvény, ahol elindítjuk a játékot.
   - private void initClasses()
     - Osztályok inicializálása.
   - private void start()
     - Thread elindítása.
   - private void updateGame()
     - GameState szerint hívja másik osztály update metódusát.
   - public void run()
     - Thread megvalósítása. Fixálva van 120 FPS és 60 UPS (Ez utóbbi a játékmenet updatelésének mennyisége másodpercenként)
   - public Render getRender()
     - Visszaadja a render-t.
   - public Menu getMenu()
     - Visszaadja a menu-t.
   - public Playing getPlaying()
     - Visszaadja a kiválasztott szinthez tartozó playing-et.
   - public Levels getLevels()
     - Visszaadja a levels-t.
   - public GameOver getGameOver()
     - Visszaadja a gameOver-t.
   - public GameWon getGameWon()
     - Visszaadja a gameWon-t.
   - public int getLvlsDone()
     - Visszaadja az elért szintet.
   - public void setLvlsDone(int lvl)
     - Beállítja az elért szintet a paraméterként kapott értékre, ha annál a jelenlegi kisebb vagy egyenlő.
   - public TileHandler getTileHandler()
     - Visszaadja tileHandlert.
   - public int getSelectedLevel()
     - Visszaadja a kiválasztott szintet.
   - public void setSelectedLevel(int id)
     - Beállítja a kiválasztott szintet a paraméterként kapott értékre.

20.GameScreen:

1. **Leírás:**
   - A játék panelját megvalósító osztály. Leszármazik a JPanel-ből.
2. **Változók:**
   - private static final long serialVersionUID = 1L;
   - private Game game;
   - private Dimension size;
   - private MyMouseListener myMouseListener;
3. **Metódusok:**
   - public GameScreen(Game game)
     - Konstruktor.
   - public void initInputs()
     - Input inicializálása.
   - private void setPanelSize()
     - Panel méretének beállítása és fixálása
   - public void paintComponent(Graphics g)
     - Meghívja a JPanel paintComponent metódusát majd renderel.

21. Render:

1. **Leírás:**
   - Renderelést megvalósító főosztály.
2. **Változók:**
   - private Game game;
3. **Metódusok:**
   - public Render(Game game)
     - Konstruktor.
   - public void render(Graphics g)
     - GameState-től függően hívja meg más-más osztályok render metódusát.

22. GameState (enum):

1. **Leírás:**
   - GameState-ek ennumerációja. Ezzel döntjük el, hogy éppen melyik képernyőr jelenítjük meg.
2. **Ennumeráció:**
   - PLAYING
   - MENU
   - LEVELS
   - GAMEOVER
   - GAMEWON
3. **Változók:**
   - public static GameStates gameState = MENU;
4. **Metódusok:**
   - public static void setGameState(GameStates state)

- Beállítja a gameState-et a paraméterként kapottra.

23.GameScene:

1. **Leírás:**
   - Különböző játékon belüli képernyők ősosztálya.
2. **Változók:**
   - protected Game game;
3. **Metódusok:**
   - public GameScene(Game game)
     - Konstruktor.
   - public Game getGame()
     - Visszaadja a game-t.

24.SceneMethods (Interfész):

1. **Leírás:**
   - GameScene-ekhez tartozó interfész.
2. **Változók:**
   - -
3. **Metódusok:**
   - public void render(Graphics g);
     - Renderelés.
   - public void mouseClicked(int x, int y);
     - Ha kattintunk akkor csinál valamit.
   - public void mouseMoved(int x, int y);
     - Ha az egér mozog akkor csinál valamit.
   - public void mousePressed(int x, int y);
     - Ha az egeret lenyomjuk akkor csinál valamit.
   - public void mouseReleased(int x, int y);
     - Ha az egeret elengedjük csinál valamit.
   - public void mouseDragged(int x, int y);
     - Ha az egeret húzzuk csinál valamit.

25.Menu:

1. **Leírás:**
   - Menüt valósítja meg. Leszármazik a GameScene osztályból és implementálja a SceneMethods interfészt.
2. **Változók:**
   - private CustomButton buttonPlay, buttonQuit;
3. **Metódusok:**
   - public Menu(Game game)
     - Konstruktor.
   - private void initButtons()

        - Inicializálja a gombokat.
    - public void render(Graphics g)
        - Renderelés megvalósítása.
    - private void drawButtons(Graphics g)
        - Kirajzolja a gombokat.
    - public void mouseClicked(int x, int y)
        - Gombokra kattintás esetén csinál valamit.
    - public void mouseMoved(int x, int y)
        - Ha gomb felett mozog az egerünk akkor csinál valamit.
    - public void mousePressed(int x, int y)
        - Ha megnyomunk egy gombot akkor csinál valamit.
    - public void mouseReleased(int x, int y)
        - Ha elengedjük az egeret akkor meghívja a resetButtons metódust.
    - private void resetButtons()
        - Visszaállítja a gombok boolean értékeit.

26. Levels:

1. **Leírás:**
    - A szint menü megjelenítését valósítja meg. Leszármazik a GameScene osztályból és implementálja a SceneMethods interfészt.
2. **Változók:**
    - private ArrayList<CustomButton> levelButtons = new ArrayList<>();
    - private CustomButton menuButton;
3. **Metódusok:**
    - public Levels(Game game)
        - Konstruktor.
    - private void initButtons()
        - Gombok inicializálása.
    - public void render(Graphics g)
        - Renderelés.
    - private void drawButtons(Graphics g)
        - Kirajzolja a gombokat.
    - public void mouseClicked(int x, int y)
        - Gombokra kattintás esetén csinál valamit.
    - public void mouseMoved(int x, int y)
        - Ha gomb felett mozog az egerünk akkor csinál valamit.
    - public void mousePressed(int x, int y)
        - Ha megnyomunk egy gombot akkor csinál valamit.
    - public void mouseReleased(int x, int y)
        - Ha elengedjük az egeret akkor meghívja a resetButtons metódust.

- private void resetButtons()
  - Visszaállítja a gombok boolean értékeit.

27. Playing:

1. **Leírás:**
   - A kiválasztott szint megjelenítését valósítja meg. Leszármazik a GameScene osztályból és implementálja a SceneMethods interfészt.
2. **Változók:**
   - private int [][] lvl;
   - private int lvlId;
   - private ArrayList<Point> path = new ArrayList<>();
   - private Tile selectedTile;
   - private Tower selectedTower;
   - private BottomBar bottomBar;
   - private int mouseX, mouseY;
   - private int money;
   - private int lives;
   - private boolean drawSelect;
   - private boolean canEdit = true;
   - private int animIdx;
   - private int tick;
   - private EnemyHandler enemyHandler;
   - private TowerHandler towerHandler;
   - private ProjectileHandler projectileHandler;
   - private WaveHandler waveHandler;
3. **Metódusok:**
   - public Playing(Game game, int lvlId)
     - Konstruktor.
   - public void saveLevel()
     - Elmenti a szintet.
   - public void loadLevel(int[][] lvl, ArrayList<Point> path, ArrayList<Tower> towers, int wave, int money, int lives)
     - Beállítja az adott szint adatait.
   - public void render(Graphics g)
     - Renderelés. Rajzoló metódusokat hív meg.
   - private void updateTick()
     - Tick-eket növel és resetel.
   - public void update()
     - Ha nem szerkesztés módban van a játék akkor updateli a waveHandler, illetve ha meghalt az összes szörny, és nem az utolsó wave volt akkor átrak szerkesztő módba és cleareli a szörnyek listáját. Ellenkező esetben ha az utolsó wave volt akkor áttob

GAMEWON state-be. Ez a metódus spawnolja le a szörnyeket is megfelelő időközönként. A végén a maradék Handler-öket is updateli.

- private boolean isLastWave()
  - Visszaadja, hogy ez volt-e az utolsó wave.
- private boolean areAllEnemiesDead()
  - Visszaadja, hogy meghalt-e az összes szörny.
- public void nextRound()
  - Meghívja a bottomBar updateTileButtons metódusát, canEditet falsera állítja, meghívja a következő hullámot, majd aszerint, hogy mennyi pénzünk van, növeli a megfelelő értékkel.
- public void removeLife()
  - Levesz egy életet. Ha 0 alá csökkenne ezután az életünk akkor GAMEOVER state-be dob.
- private void drawLevel(Graphics g)
  - Kirajzolja a szint elemeit.
- private boolean isAnim(int id)
  - Visszaadja, hogy az adott azonosítójú elem animált-e.
- private BufferedImage getSprite(int id)
  - Visszaadja az adott azonosítójú elemhez tartozó képet.
- private BufferedImage getSprite(int id, int animIdx)
  - Visszaadja az adott azonosítójú és animációazonosítójú animált elemhez tartozó megfelelő képet.
- private void drawSelectedTile(Graphics g)
  - Kirajzolja a kiválasztott elemet, ha van.
- private void drawSelectedTower(Graphics g)
  - Kirajzolja a kiválasztott tornyot, ha van.
- public void mouseClicked(int x, int y)
  - Ellenőrzi, hogy hova kattintunk, ez alapján hívhatja a bottomBar mouseClicked metódusát, lerakhat egy elemet, illetve egy tornyot, vagy meghívhatja a bottomBar displayTowerInfo metódusát.
- private void changeTile(int x, int y)
  - Ha van kiválasztott elem, akkor a megadott koordinátákra megfelelően lerakja, megváltoztatja a lvl-ben a értékét, és hozzáadja az út végéhez.
- private boolean placedRoadRight(int x, int y)
  - Visszaadja, hogy az utat jó helyre rakjuk-e.
- private boolean placedTowerRight(int x, int y)
  - Visszaadja, hogy a tornyot jó helyre rakjuk-e.
- private Tower getTowerAt(int x, int y)
  - Meghívja a towerHandler getTowerAt metódusát a megadott paraméterekkel.
- public void shootEnemy(Tower t, Enemy e)
  - Létrehoz egy új lövedéket t és e paraméterekkel.

- private void spawnEnemy()
  - Lespawnolja a következő szörnyet a hullámból.
- private boolean isTimeForSpawn()
  - Visszaadja, hogy elegendő idő telt-e el, hogy lerakhasson a program egy új szörnyet.
- public void mouseMoved(int x, int y)
  - Ellenőrzi, hogy hol van az egerünk és ez alapján meghívhatja a bottomBar mouseMoved metódusát, vagy állíthatja az egerünk helyzetét reprezentáló változókat.
- public void mousePressed(int x, int y)
  - Csak a bottomBar mousePressed metódusát hívja meg, ha ott van az egér.
- public void mouseReleased(int x, int y)
  - Ha elengedjük az egeret, meghívja a booleaneket visszaállító metódust.
- public TileHandler getTileHandler()
  - Visszaadja a tileHandler-t.
- public TowerHandler getTowerHandler()
  - Visszaadja a towerHandler-t.
- public EnemyHandler getEnemyHandler()
  - Visszaadja az enemyHandler-t.
- public WaveHandler getWaveHandler()
  - Visszaadja a waveHandler-t.
- public int getTileType(int x, int y)
  - Visszaadja a megadott koordinátákon lévő elem azonosítóját.
- public ArrayList<Point> getPath()
  - Visszaadja az út listáját.
- public int getMoney()
  - Visszaadja a pénz mennyiségét.
- public void addMoney(int m)
  - Hozzáad a pénz mennyiségéhez a paraméterként kapott értékkel.
- public int getLives()
  - Visszaadja az életünket.
- public boolean canEdit()
  - Visszaadja, hogy szerkeszthetjük-e a pályát.
- public void setSelectedTile(Tile tile)
  - Beállítja a kiválasztott elemet.
- public void setSelectedTower(Tower tower)
  - Beállítja a kiválasztott tornyot.
- public void resetLevel()
  - Visszaállítja a szintet az eredeti állapotba.
- private void resetPath()

- Visszaállítja az út listát kezdőállapotba.

28.GameWon:

1. **Leírás:**
   - A nyerés képernyőt valósítja meg. Leszármazik a GameScene osztályból és implementálja a SceneMethods interfészt.
2. **Változók:**
   - private CustomButton replayButton, menuButton;
3. **Metódusok:**
   - public GameWon(Game game)
     - Konstruktor
   - private void initButtons()
     - Inicializálja a gombokat.
   - public void render(Graphics g)
     - Renderelés. Rajzoló függvényeket hív, illetve kirajzolja a „Game Won!” szöveget.
   - private void drawButtons(Graphics g)
     - Kirajzolja a gombokat.
   - private void drawText(Graphics g, String text, int x, int y, int width, int height)
     - A megadott string-et rajzolja ki a megadott kereteken belül szimmetrikusan.
   - private void resetGame()
     - Meghívja a szintet resetelő metódust a game-en keresztül.
   - public void mouseClicked(int x, int y)
     - Ha a Menu-re kattintunk a menübe dob, ha a Replay-re akkor újrajátszhatjuk a szintet.
   - public void mouseMoved(int x, int y)
     - Ha valamelyik gomb felett van akkor máshogy rajzolja ki azt a gombot.
   - public void mousePressed(int x, int y)
     - Ha megnyomjuk valamelyik gombot, akkor máshogy rajzolja ki.
   - public void mouseReleased(int x, int y)
     - Ha elengedjük az egeret, akkor meghívja a resetButtons metódust.
   - private void resetButtons()
     - Visszaállítja a gombok booleanjait.

29.GameOver:

1. **Leírás:**
   1. A vesztés képernyőt valósítja meg. Leszármazik a GameScene osztályból és implementálja a SceneMethods interfészt.
2. **Változók:**
   1. private CustomButton replayButton, menuButton;
3. **Metódusok:**
   1. public GameWon(Game game)
      1. Konstruktor

2. private void initButtons()
    1. Inicializálja a gombokat.
3. public void render(Graphics g)
    1. Renderelés. Rajzoló függvényeket hív, illetve kirajzolja a „Game Over!” szöveget.
4. private void drawButtons(Graphics g)
    1. Kirajzolja a gombokat.
5. private void drawText(Graphics g, String text, int x, int y, int width, int height)
    1. A megadott string-et rajzolja ki a megadott kereteken belül szimmetrikusan.
6. private void resetGame()
    1. Meghívja a szintet resetelő metódust a game-en keresztül.
7. public void mouseClicked(int x, int y)
    1. Ha a Menu-re kattintunk a menübe dob, ha a Replay-re akkor újrajátszhatjuk a szintet.
8. public void mouseMoved(int x, int y)
    1. Ha valamelyik gomb felett van akkor máshogy rajzolja ki azt a gombot.
9. public void mousePressed(int x, int y)
    1. Ha megnyomjuk valamelyik gombot, akkor máshogy rajzolja ki.
10. public void mouseReleased(int x, int y)
    1. Ha elengedjük az egeret, akkor meghívja a resetButtons metódust.
11. private void resetButtons()
    1. Visszaállítja a gombok booleanjait.

30. MyMouseListener:

1. **Leírás:**
    - Saját MouseListener.
2. **Változók:**
    - private Game game;
3. **Metódusok:**
    - public MyMouseListener(Game game)
        - Konstruktor.
    - public void mouseDragged(MouseEvent e)
        - Nem csinál semmit
    - public void mouseMoved(MouseEvent e)
        - GameState alapján hívja más osztályok mouseMoved metódusát.
    - public void mouseClicked(MouseEvent e)
        - GameState alapján hívja más osztályok mouseClicked metódusát.
    - public void mousePressed(MouseEvent e)
        - GameState alapján hívja más osztályok mousePressed metódusát.
    - public void mouseReleased(MouseEvent e)
        - GameState alapján hívja más osztályok mouseReleased metódusát.
    - public void mouseEntered(MouseEvent e)

    - Nem csinál semmit
- public void mouseExited(MouseEvent e)
    - Nem csinál semmit

3. Felhasználói dokumentáció:

• A játék leírása:

- A Builder Tower Defense játék alapja egy sima tower defense játék:
  - Egy úton mennek végig szörnyek különböző hullámokban és meg kell akadályoznunk különböző tornyokkal (lőnek, lassítanak stb.), hogy eljussanak az út végéig. Ebben a játékban a játékos építi meg az utat is egy előre megszabott korlátozott területen.
  - Új játék kezdésekor egy négyzetekre osztott fix pályát kapunk egy előre lehelyezett 5 négyzet hosszú úttal és egy toronnyal, ezt az utat kell bővítenünk. Az út végét egy piros pötty jelzi, melyet bővíthetünk. Az út eleje, ahonnan a szörnyek indulnak pedig a másik vége, errefelé nem lehet bővíteni az utat.
  - A játékosnak 10 szíve van, ha egy szörny bemegy az -1 szív. Ha elfogy az összes szív akkor a játékot elvesztettük.
  - A játékosnak van pénze melyet szörnyek megölésekor random kaphat, illetve minden hullám végén növekszik attól függően, hogy mennyi pénze van összesen a játékosnak (10 alatt +1, 10-től 20-ig +2, 30-tól +3).
  - Hullámok között van időnk építkezni, itt jönnek be az extra elemek a játékba. 2 rublikában kapjuk a lerakható dolgokat, az elsőt lerakhatjuk ingyen a másodikat pedig a következő kör elején lesz ingyen elérhető, de meg is vásárolhatjuk, hogy elérhető legyen 2 pénzért, ekkor a megvásárolt elem kerül előre és a második helyen új lerakható dolog jelenik meg. Nem muszáj mindent letennünk, de ha valamit nem teszünk le az első helyről és a következő kör jön vagy megvásároltuk a mögötte lévőt, akkor az első helyen lévő nem lerakott dolog eltűnik. Egy körben akárhányszor vásárolhatunk amíg van rá pénzünk.
  - A játékos a 2 rublikában randomizálva kaphat torony és út elemeket, ugyanis ahogy azt az elején említettük a játékosnak kell megépítenie az utat is. Utat és tornyot csak másik út mellé rakhatunk le. Az út nem fordulhat vissza magába.
  - Egy szint akkor ér véget, ha az összes hullám lement, ekkor nyertünk és elérhetővé válik a következő pálya.

• A játék kezelése:

  - Indításkor 2 lehetőségünk van: **Play**, **Quit**. A Quit lehetőséggel léphetünk ki, a Play lehetősség pedig a szintválasztáshoz dob.
  - 10 szint van a játékban, a nem elérhető szintek pirossal át vannak húzva. Az elején csak az első érhető el, és innen haladhatunk tovább. Ha egy szintet sikeresen végigjátszunk akkor elérhetővé válik a következő.
  - A szintet kiválasztva az elején még van lehetőségünk egy új elemet lerakni.
  - Elem lerakásakor a megjelenő **Cancel** gombbal tudjuk visszavonni az elem kiválasztását.
  - A hullámot a **Next round** gombbal indíthatjuk el.
  - Új elemet a **Next tile** gombbal vehetünk.
  - Ezen gombok mellett jelzi a játék, hogy mennyi pénzünk, illetve mennyi életünk van.
  - Alattuk pedig, hogy hanyadik hullám megy most.
  - Új hullámot indítva a lenti sáv tetején írja, hogy mennyi szörny van még életben.
  - Egy pályán lévő toronyra rákattintva alapvető tulajdonságokat és a hatótávolságát tudhatjuk meg.

- A **Save** gombbal menthetjük el a játék állását, a **Menu** gombbal pedig visszaléphetünk a menübe.
- Ha egy szintet végigjátszunk vagy elvesztünk, akkor a megjelenő képernyőn is két lehetőségünk van: **Replay**, **Menu**. A Replay gombbal újrajátszhatjuk a szintet, a Menu gombbal kiléphetünk a menübe.