SZOFTVERTECHNOLÓGIA ÉS GRAFIKUS FELHASZNÁLÓI INTERFÉSZ TERVEZÉSE

CSOPORTMUNKA:
EGY KLASSZIKUS TOWER-DEFENCE JÁTÉK

Készítették:

- **Deák László Szilamér**
- **Szeiler Simon**

TARTALOMJEGYZÉK

A játék rövid leírása:

Célunk egy klasszikus tower-defence játék megvalósítása. A játék során egy várat védünk, amelyet támadó seregek egyre erősödő hullámban ostromolnak. Kezdetben védelmünk nincs, azt a játék során építjük ki alkalmazkodva a támadó seregekhez.

Az alap játékmód bemutatása:

Egy játék során a támadó lények végtelen számú seregével állunk szemben. A játékban a cél minél tovább kitartani és minél több pontot összegyűjteni. A játék bemutathatósága érdekében, ha kitartunk addig, amíg 110. ellenség is megjelenik, a játék befejeződik a játékos győzelmével. Az ellenfelek elpusztításával erőforrást gyűjthetünk, amellyel védekező objektumokat finanszírozhatunk meg. A támadó seregek ereje és érkezésük intenzitása kizárólag időhöz és egy kis véletlenfaktorhoz kötött.
A védekező egységek elhelyezése tetszőleges, számuk azonban korlátozott. A támadó egységek egy kötött útvonalat járnak be és kizárólag a páncélzatuk és az életerejük védi a védekező tornyok ellen.
A játék során a védekező oldalt képviseljük, így lehetőségünk van

1) tetszőleges füves mezőre tornyot telepíteni, amely típusát mi határozzuk meg gombnyomással vagy a jobboldali listából kattintással
2) a tornyot fejleszteni
3) tornyot bontani

A fejlesztés első körben a torony önmagára való újbóli lehelyezésével valósítható meg, aminek hatására a torony tulajdonságai javulnak.

Adminisztráció:

A játék során eredményünket fileban fogjuk tárolni. Ennek megfelelően minden játék indításakor meg kell adnunk a játékbéli nevünket. Ezek alapján, a képernyőn eddigi tíz legtöbb pontot elérő játékosnevet listázni fogjuk.
A játék során annak bezárásakor az aktuális állapot tárolódik egy, a játékbeli nevünkhöz rendelt fileban. Állás visszatöltésekor ez az állás kerül visszatöltésre.

A játék részletes leírása:

A projekt megvalósítása során a védekező játékost az építhető tornyok számával (6 darab) és az egyszerre rendelkezésre álló nyersanyag mennyiségével korlátozzuk (ez induláskor 1500).
Hat fajta toronyból választhatunk, melyek eltérő sebzés típussal küzdenek a támadók ellen. Ezek sorban a következők:

Sebzés típus	Alapár	Sebzés jellemzése
Fizikai	250	Az alap sebzést az ellenfél páncélja csökkenti
Méreg	900	Az alap sebzésen felül másodpercenként az elsőző sebzése felét újra kiosztja, ameddig a sebzés legalább 2
Tűz	260	Az alap sebzését kiosztja a páncélt figyelmen kívül hagyva
Fagy	270	Az alap sebzés páncéllal csökkentett értékének 80%-át sebzi és az alap sebzés 20%-ával csökkenti a mozgási sebességét az ellenfélnek
Levegő	250	Az alap sebzés páncéllal csökkentett értékének 20%-át sebzi és az ellenfelet egy pszeudo-véletlen irányba fújja, ahonnan az ellenfélnek vissza kell térnie eredeti helyére, a támadás folytatásához
Föld	270	Az alap sebzés páncéllal csökkentett értékének 80%-át sebzi és az alap sebzés 20%-ával csökkenti az ellenfél páncélzatát

A megjelenő ellenfelek ereje sebessége és páncélzata az egy adott intervallumon belül véletlenszerűen váltakozik. Ennek megfelelően az életerő 50-99, a páncélzat 2-9, a sebesség 3-6 között váltakozik. A játék során

időnként (6% eséllyel) a szokásos ellenfeleken felül megidéz egy kivételesen erős ellenfelet. Ennek életereje 150-297, páncélzata 10-24 között váltakozik, sebességük 3. A játék során minden tizedik ellenfél megjelenése után gyorsul egy kicsit az új ellenfelek megjelenésének sebessége.
A játék addig tart, ameddig a 110. ellenfél meg nem jelenik vagy a 10. ellenfél be nem ér a célterületre, azaz ameddig végig nem megy az útvonalon.

Model és Logic:

TowerDefenseModel

```
private readonly List<Enemy> enemies = new List<Enemy>();
private readonly List<Tower> towers = new List<Tower>();
private readonly List<Projectile> projectiles = new List<Projectile>();

public List<Enemy> Enemies { get { return enemies; } }
public List<Tower> Towers { get { return towers; } }
public List<Projectile> Projectiles { get { return projectiles; } }
public TowerSelectorRect[] TowerSelectorRects { get; set; }
public bool[,] Fields { get; set; }
public bool[,] Path { get; set; }
public double GameWidth { get; set; }
public double GameHeight { get; set; }
public double TileSize { get; set; }
public Point EntryPoint { get; set; }
public Point ExitPoint { get; set; }
public int Coins { get; set; }
```

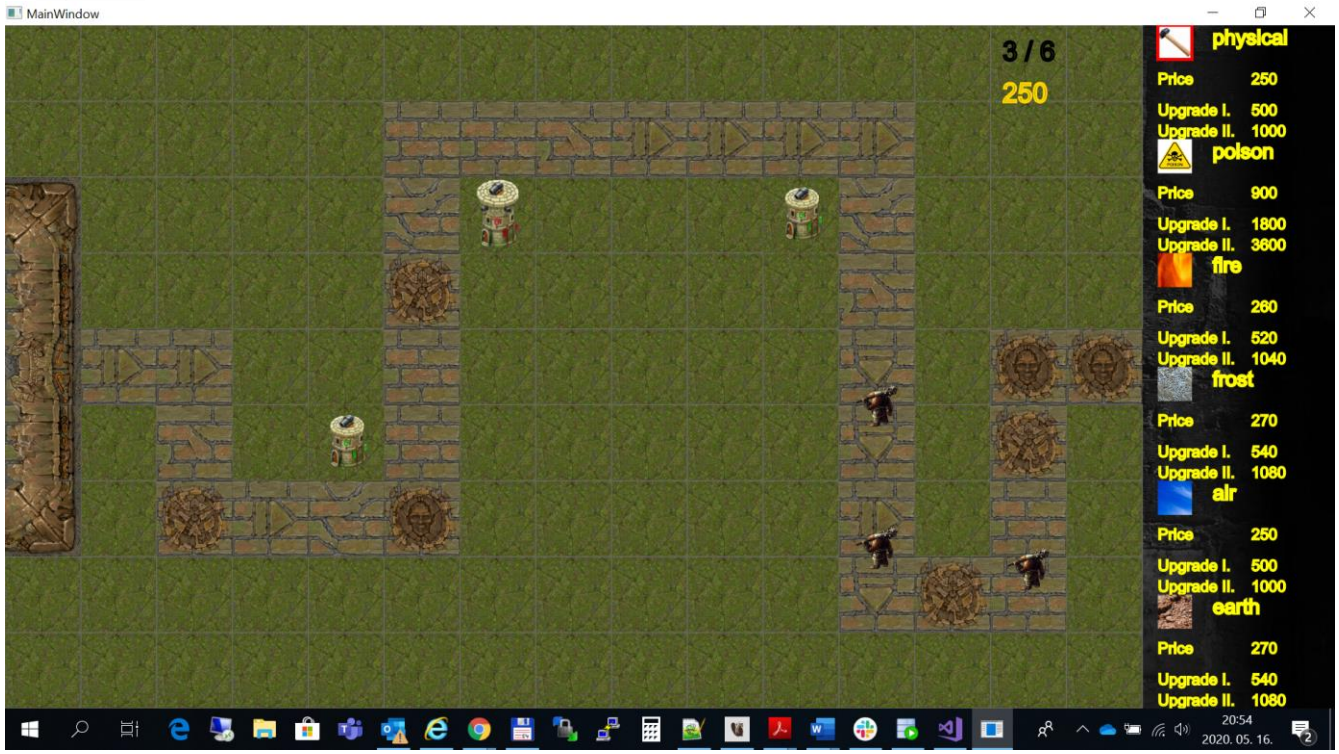
TowerDefenseLogic

```
public int baseTickSpeed = 40;
public int enemyCounter = 0;
public int playerHealth;
public Action finishGame;
List<Enemy> deleteEnemies = new List<Enemy>();
TowerDefenseModel model;
private string userName;

public void MoveEnemies(List<Enemy> enemyList)
public int SpawnNewEnemy(Action raiseSpawnSpeed)
public void MoveProjectiles(List<Projectile> projList)
private Point SetNewDestionation(MovingGameItem enemy)
public void SetTowerTargets(List<Enemy> enemyList, List<Tower> towerList)
public bool AddOrUpgradeTower(Point mousePos, DispatcherTimer timer)
private bool AddTower(Point mousePos, DispatcherTimer timer)
private bool UpgradeTower(Tower choosenTower)
public bool RemoveTower(Point mousePos)
private Tower ExistsTower(Point mousePos)
public void Framing(Point mousePos)
public TowerSelectorRect GetSelectedTower()
public Point GetTilePos(Point mousePos)
public Point GetPosTile(Point tile)
public Point GetPosTileCentre(Point tile)
private void SetPath(bool[,] path)
```

Menu wireframe:

Game wireframe:



Class diagram: