# Dokumentáció

## Mozgás

A mozgás a billentyűzeten lévő nyilak segítségével fog működni. Két dimenziós térben lehet elképzelni a mozgást. Fel, Le, Jobbra, Balra lehet majd mozogni a pálya területén belül.

A gombok bevitelét követően azonnal történni fog a mozgás.

Alap logika:

1. public virtual void ProcessInput(ConsoleKeyInfo keyInfo)
2. {
3. switch (keyInfo.Key)
4. {
5. case ConsoleKey.LeftArrow:
6. CurrentLevel.moveEntity(0, CurrentLevel.entities[0].posX - 1, CurrentLevel.entities[0].posY);
7. break;
8. case ConsoleKey.RightArrow:
9. CurrentLevel.moveEntity(0, CurrentLevel.entities[0].posX + 1, CurrentLevel.entities[0].posY);
10. break;
11. case ConsoleKey.DownArrow:
12. CurrentLevel.moveEntity(0, CurrentLevel.entities[0].posX, CurrentLevel.entities[0].posY + 1);
13. break;
14. case ConsoleKey.UpArrow:
15. CurrentLevel.moveEntity(0, CurrentLevel.entities[0].posX, CurrentLevel.entities[0].posY - 1);
16. break;
17. }
18. if (IsLevelCompleted())
19. {
20. HandleLevelCompletion();
21. }
22. }

## Jelenet kezelő

Ez egy olyan osztály lesz ami kezelni fogja, hogy éppen milyen jelenet látható a konzolos applikációban. Pl.: Bevezető, majd innen átvált a Főmenüre, Főmenüről az első szintre, és így tovább.

Továbbá ez fogja feldolgozni a Mozgást is.

1. namespace Rise\_of\_Derma.scenarios
2. {
3. class Scenario
4. {
5. public Level CurrentLevel { get; set; }
6. public Scenario()
7. {
8. CurrentLevel = GenerateScenario();
9. }
10. // Virtuális szint, felülírja majd a szint generáló
11. public virtual Level GenerateScenario()
12. {
13. Level level = new Level();
14. return level;
15. }
16. public virtual void ProcessInput(ConsoleKeyInfo keyInfo)
17. {
18. // Bemenetek feldolgozása
19. }
20. protected virtual bool IsLevelCompleted()
21. {
22. // Szint teljesítettségének ellenőrzése
23. return false;
24. }
25. protected virtual void HandleLevelCompletion()
26. {
27. // Szint teljesítésének lekezelése
28. }
29. }
30. }

## Entitás osztály

Ez tartalmazza az összes fajta entitás tulajdonságait. Életerő, hogy néz ki a konzolon, szint, pozíció, előző pozíció. Tartalmazza az eljárásokat az entitás tulajdonságainak befolyásolásához is.

1. namespace Rise\_of\_Derma.entities
2. {
3. class Entity
4. {
5. public int health = 100;
6. public char appearance;
7. public int power = 1;
8. public int prevPosX;
9. public int prevPosY;
10. public int posX;
11. public int posY;
12. public void damage(int amount)
13. {
14. health -= amount;
15. }
16. public void regenerate(int amount)
17. {
18. health += amount;
19. }
20. public void setHealth(int amount)
21. {
22. health = amount;
23. }
24. public void moveEntityProperties(int prevPosX, int prevPosY, int posX, int posY)
25. {
26. this.prevPosX = prevPosX;
27. this.prevPosY = prevPosY;
28. this.posX = posX;
29. this.posY = posY;
30. }
31. }
32. }

Játékos osztály

Az entitás osztály alosztálya. A játékos tulajdonságait írja le. Például a kinézetét.

1. namespace Rise\_of\_Derma.entities
2. {
3. class Player : Entity
4. {
5. public int crystalCount = 0;
6. public Player()
7. {
8. appearance = 'X';
9. }
10. }
11. }

## Ellenség osztály

Szintén az entitás osztály alosztálya. Ez az ellenségek tulajdonságait írja le.

1. namespace Rise\_of\_Derma.entities
2. {
3. class Enemy : Entity
4. {
5. public Enemy()
6. {
7. appearance = 'O';
8. }
9. }
10. }

## Térkép osztály

A játék terület 20x80 lesz, az ablak mérete pedig 24x82 a keretek és egyéb adatok megjelenítésének céljából. Tartalmazza azokat az adatokat amik a szint létrehozásához szükségesek.

1. namespace Rise\_of\_Derma.fields
2. {
3. class Map
4. {
5. // Size of the map
6. public int h = 20;
7. public int w = 80;
8. }
9. }

Szint osztály

Ez az osztály a szinteket fogja kezelni, azaz a szintek elemeit, és azt hogy a különféle szintek miből épülnek fel. Például hol vannak a játékosok, a falak és az ellenségek. (Térképnek az alosztálya)

## Szöveges információk

Ablak neve: Rise of Derma

Szövegek: Szint, Idő, Életerő, Harcrendszer

A játékos szintet léphet ha egy ellenséget kivégez.

Egy ellenség kivégzéséhez a játékosnak nagyobb szintűnek kell lennie, mint egy ellenségnek. A játékos onnan tájékozódhat egy ellenség szintjéről, hogy meg van adva melyik karakter mennyit ér.

## Tárgyak

Kristály: Felszedhető

Fal: Nem lehet rálépni, meg kell kerülni, nem lehet elpusztítani

Példa:

1. namespace Rise\_of\_Derma.entities.constants
2. {
3. class Wall : Entity
4. {
5. public Wall()
6. {
7. appearance = '#';
8. }
9. }
10. }

## Pontrendszer

A játékos a pontokat úgy szeri, hogy minél rövidebb idő alatt teljesíti a játékot, annál több pontot kap.

Ezt majd kiszámolja a program és szinkronizálja a szerverrel, hogy a ranglistára felkerüljön.

A szerverrel való kommunikáció HTTP Rest API POST requestekkel fog történni. Ezt egy egyszerű kis backend fogja kezelni.