**2d-s Platformer**

Témalabor

# A Játék leírása

A témalabor során egy 2d-s platformert készítettem el a MonoGame nevű keretrendszerben. A játékosnak a karakterét irányítva kell eljutnia a célig. Az „a” és a „d” billentyűk megnyomásával a karakter oldalra tud mozogni. A szóköz billentyűvel tud a karakter ugrani. Az „n” és az „m” billentyűkkel tud a karakter támadni. Amennyiben a játékost megüti egy ellenség, vagy belesétál egy csapdába, úgy veszít az életéből. Amennyiben a játékos élete elfogy, úgy meghal és végetér a játék. A játékosnak van lehetősége a pályán különböző érméket összegyűjteni, ezzel plusz pontokat szerezve, a pálya végigjátszása során.

# Components

A Components névtérben találhatók, a különböző komponenseket, például gombokat kirajzoló és kezelő osztályok.

## RectangleDrawer.cs

Ez az osztály téglalapok kirajzolását végzi el.

**Metódusok:**

* DrawRectangle(): Statikus függvény, ami kirajzol egy téglalapot a megadott paraméterek alapján.
  + SpriteBatch spriteBatch: A SpriteBatch, amin a rajzolás történik.
  + Rectangle rectangle: A kirajzolandó téglalap.
  + Color color: A téglalap színe.
  + int lineWidth: A téglalap szélének vastagsága.
* DrawRectangleWithBackground(): Statikus függvény, ami kirajzol egy téglalapot háttérszínnel a megadott paraméterek alapján.
  + SpriteBatch spriteBatch: A SpriteBatch, amin a rajzolás történik.
  + Rectangle rectangle: A kirajzolandó téglalap.
  + Color color: A téglalap színe.
  + int lineWidth: A téglalap szélének vastagsága.
  + Color backgroundColor: A téglalap háttérszíne.

## Component.cs

Abstract osztály, amiből leszármaznak a különböző komponensek.

**Metódusok:**

* Update(): Abstract függvény. A komponens frissítését végzi el.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.
* Draw(): Abstract függvény. A komponens kirajzolását végzi el.
  + SpriteBatch spriteBatch: A SpriteBatch, amin a rajzolás történik.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.

## Button.cs

A gombokat kezelő osztály. Leszármazik a Component osztályból. Kirajzolja az adott gombot és kattintás esetén meghívja annak eseménykezelő függvényeit.

**Metódusok:**

* Button(): A konstruktor a gomb attribútúmainak beállításáért felelős.
  + SpriteFont font: A gomb feliratának fontja.
  + string text: A gomb felirata.
  + Vector2 position: A gomb pozíciója.
  + Vector2 size: A gomb mérete
  + Color color: A gomb színe.
  + Color colorHovering: A gomb színe, amennyiben az egér fölé kerül.
* Update(): Megváltoztatja a gomb színét, amennyiben az egér fölé került, valamint meghívja a gomb eseménykezelő függvényeit, amennyiben rá kattint a felhasználó.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.
* Draw(): A gomb kirajzolását végzi el.
  + SpriteBatch spriteBatch: A SpriteBatch, amin a rajzolás történik.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.

# Collision

A Collision névtérben található a collision boxokat kezelő osztályok. A játék ezeket használja az objektumok ütközésének érzékeléséhez.

## CollisionBox.cs

CollisionBoxok reprezentálásáért felelős osztály.

**Metódusok:**

* CollisionBox(): A konstruktor beállítja a collision box attribútumait.
  + Vector2 center: A collision box közepének a pozíciója.
  + Vector2 halfSize: A collision box méreteinek a fele.
* IsCollided(): Ellenőrzi, hogy a paraméterben megadott másik collision boxal ütközik-e az objektum.
  + CollisionBox other: A másik collision boksz, amire az ütközést vizsgálni kell.

## CollisionBoxManager.cs

Eltárolja és kezeli a collision boxokat.

**Metódusok:**

* Reset (): Törli az eltárolt collision boxokat.
* AddPlayerCollisionBox(): Eltárolja egy karakternek a collision boxát.
  + CollisionBox collisionBox: Eltárolandó collision box.
  + PlayerCharacter character: A karakter, amihez a collision box tartozik.
* AddEnemyCollisionBox(): Eltárolja egy ellenségnek a collision boxát.
  + CollisionBox collisionBox: Eltárolandó collision box.
  + EnemyCharacter character: Az ellenség, amihez a collision box tartozik.
* AddGameEndCollisionBox(): Eltárolja egy játék vége elemnek a collision boxát.
  + CollisionBox collisionBox: Eltárolandó collision box.
* AddMapCollisionBox(): Eltárolja egy térképelemnek a collision boxát.
  + CollisionBox collisionBox: Eltárolandó collision box.
* RemovePlayerCollisionBox(): Törli egy karakternek a collision boxát.
  + CollisionBox collisionBox: Törlendő collision box.
* RemoveEnemyCollisionBox(): Törli egy ellenségnek a collision boxát.
  + CollisionBox collisionBox: Törlendő collision box.
* RemoveGameEndCollisionBox(): Törli egy játék vége elemnek a collision boxát.
  + CollisionBox collisionBox: Törlendő collision box.
* RemoveMapCollisionBox(): Törli egy térképelemnek a collision boxát.
  + CollisionBox collisionBox: Törlendő collision box.
* IntersectWithMap(): Ellenőrzi egy térképelemmel való ütközést.
  + CollisionBox collisionBox: Ellenörzendő collision boksz.
* IntersectWithPlayer(): Ellenőrzi egy karakterrel való ütközést.
  + CollisionBox collisionBox: Ellenörzendő collision boksz.
* IntersectWithEnemy(): Ellenőrzi egy ellenséggel való ütközést.
  + CollisionBox collisionBox: Ellenörzendő collision boksz.
* IntersectWithGameEndBoxes(): Ellenőrzi egy játék vége elemmel való ütközést.
  + CollisionBox collisionBox: Ellenörzendő collision boksz.
* Draw(): Kirajzolja az összes collision boxot.
  + SpriteBatch spriteBatch: A SpriteBatch, amin a rajzolás történik.
* DrawCollisionBox(): Kirajzol egy collision boxot a paraméterek alapján.
  + SpriteBatch spriteBatch: A SpriteBatch, amin a rajzolás történik.
  + CollisionBox collisionBox: A kirajzolandó collision box.
  + Color color: A rajzolás színe.

# Texture

A Texture névtérben találhatók a textúrák kirajzolását segítő osztályok. A SpriteSheet osztály a spritesheetek feldolgozását végzi el, míg az Animation osztály a spritesheet animációk lejátszásáért felelős.

## SpriteSheet.cs

Egy spritesheetet reprezentál.

**Metódusok:**

* SpriteSheet(): A konstruktor beállítja a spritesheet paramétereit.
  + Texture2D texture: A spritesheet textúrája.
  + int spriteXCount: A spritesheet oszlopainak száma.
  + int spriteYCount A spritesheet sorainak száma.
* Draw(): Kirajzolja a megadott sorban és oszlopban található elemet.
  + SpriteBatch spriteBatch: A SpriteBatch, amin a rajzolás történik.
  + Vector2 position: Rajzolás pozíciója.
  + int posX: A kirajzolandó oszlop száma.
  + int posY: A kirajzolandó sor száma.
  + float scale: A nagyítás mértéke.
  + SpriteEffects effect: A rajzolás effectje.

## Animation.cs

Egy animáció lejátszásáért felelős.

**Metódusok:**

* Animation(): A konstruktor beállítja az osztály attribútumait.
  + SpriteSheet spriteSheet: Az animáció spritesheete.
  + bool repeat: Az animáció ismétlődését határozza meg.
  + double animationTime: Az animáció hossza.
* ResetAnimation (): Újraindítja az animációt.
  + SpriteBatch spriteBatch: A SpriteBatch, amin a rajzolás történik.
* NewSpriteSheet(): Betölt egy új spritesheet-et.
  + SpriteSheet spriteSheet: Betöltendő spritesheet.
* Update(): Frissíti az animációt.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.
* Draw(): Kirajzolja az animáció aktuális képkockáját.
  + SpriteBatch spriteBatch: A SpriteBatch, amin a rajzolás történik.
  + Vector2 position: A rajzolás pozíciója.

# Characters

A Characters névtérbe tartoznak a karakterek kezelésével kapcsolatos osztályok.

## CharacterDirection.cs

A karakter állapotokat felsoroló enum.

## CharacterState.cs

A karakter irányait felsoroló enum.

## Character.cs

A karaktert reprezentáló absztrakt osztály.

**Metódusok:**

* Character(): A konstruktor beállítja az attribútumokat.
  + MapManager mapManager: A mapmanager, amiben a karakter található.
* Update(): Frissíti a karaktert.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.
* Draw(): Kirajzolja a karaktert.
  + SpriteBatch spriteBatch: A SpriteBatch, amin a rajzolás történik.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.
* SetUpRunSeed(): Beállítja a karakter futási sebességét.
  + float walkSpeed: A karakter sebessége.
* AddGravity(): Beállítja a gravitációt.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.
* UpdatePhysics(): Frissíti a karakter fizikáját.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.
* Hit(): Sebzi a karaktert.
  + int damage: A sebzés mértéke.
* ActionDeath(): Absztrakt függvény. A halál állapotba váltást végzi el.
* ActionIdle(): Absztrakt függvény. Az idle állapotba váltást végzi el.
* ActionFall(): Absztrakt függvény. A zuhanás állapotba váltást végzi el.
* ActionTakeHit(): Absztrakt függvény. A sebződés állapotba váltást végzi el.

## PlayerCharacter.cs

A játékos karakterét reprezentáló absztrakt osztály. Leszármazik a Characterből.

**Metódusok:**

* Character(): A konstruktor beállítja az attribútumokat.
  + MapManager mapManager: A mapmanager, amiben a karakter található.
* ActionJump(): Absztrakt függvény. Az ugrás állapotba váltást végzi el.
* ActionRun(): Absztrakt függvény. A futás állapotba váltást végzi el.
* ActionAttack1(): Absztrakt függvény. A támadás1 állapotba váltást végzi el.
* ActionAttack2(): Absztrakt függvény. A támadás2 állapotba váltást végzi el.

## PlayerCharacterType.cs

A játékos karakterének tipusait felsoroló enum.

## Huntress.cs

A Huntress típusú játékos karaktert reprezentáló absztrakt osztály. Leszármazik a PlayerCharacterből.

**Metódusok:**

## EnemyCharacter.cs

Az ellenség karakterét reprezentáló absztrakt osztály. Leszármazik a Characterből.

**Metódusok:**

* Character(): A konstruktor beállítja az attribútumokat.
  + MapManager mapManager: A mapmanager, amiben a karakter található.
* ActionRun(): Absztrakt függvény. A futás állapotba váltást végzi el.
* ActionAttack(): Absztrakt függvény. A támadás állapotba váltást végzi el.

## Skeleton.cs

A Skeleton típusú ellenséges karaktert reprezentáló absztrakt osztály. Leszármazik a EnemyCharacterből.

**Metódusok:**

## PlayerCharacterController.cs

A játékost vezérlő osztály.

**Metódusok:**

* PlayerCharacterController(): A konstruktor beállítja az attribútumokat.
  + PlayerCharacter character: Az játékos, amihez a kontroller tartozik.
* Update(): Frissíti a kontrollert.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.
* Draw(): Kirajzolja a kontrollerhez tartozó karaktert.
  + SpriteBatch spriteBatch: A SpriteBatch, amin a rajzolás történik.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.

## EnemyCharacterController.cs

Az ellenséget vezérlő osztály.

**Metódusok:**

* EnemyCharacterController(): A konstruktor beállítja az attribútumokat.
  + EnemyCharacter character: Az ellenség, amihez a kontroller tartozik.
  + float minXPos: Az ellenség járőrözésének minimális pozíciója.
  + float maxXPos: Az ellenség járőrözésének maximális pozíciója.
* Update(): Frissíti a kontrollert.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.
* Draw(): Kirajzolja a kontrollerhez tartozó ellenséget.
  + SpriteBatch spriteBatch: A SpriteBatch, amin a rajzolás történik.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.

# Items

A Items névtérbe tartoznak a felvehető tárgyak kezelésével kapcsolatos osztályok.

## Item.cs

Abstract osztály, amiből leszármaznak a különböző felvehető tárgyak.

**Metódusok:**

* Item(): A konstruktor beállítja az attribútumokat
  + MapManager mapManager: A mapmanager, amiben az tárgy található.
* Update(): Abstract függvény. A tárgy frissítését végzi el.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.
* Draw(): Abstract függvény. A tárgy kirajzolását végzi el.
  + SpriteBatch spriteBatch: A SpriteBatch, amin a rajzolás történik.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.

## Coin.cs

Egy érme reprezentálásáért felelős.

**Metódusok:**

* Item(): A konstruktor beállítja az attribútumokat és meghívja az ős osztályt. Beállítja a collision boxot és az animációt.
  + MapManager mapManager: A mapmanager, amiben az tárgy található.
  + Vector2 pos: Az érme pozíciója.
  + int value: Az érme értéke.
* Update(): Frissíti az érme animációját. Ellenőrzi, hogy a karakter hozzáér-e az érméhez. Ha a karakter hozzáér az érméhez akkor jóváírja neki a pontot, valamint törölteti magát a térképről.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.
* Draw(): Az érme kirajzolását végzi el.
  + SpriteBatch spriteBatch: A SpriteBatch, amin a rajzolás történik.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.

# Traps

A Traps névtérbe tartoznak a csapdák kezelésével kapcsolatos osztályok.

## Trap.cs

Abstract osztály, amiből leszármaznak a különböző csapdák.

**Metódusok:**

* Update(): Abstract függvény. A csapda frissítését végzi el.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.
* Draw(): Abstract függvény. A csapda kirajzolását végzi el.
  + SpriteBatch spriteBatch: A SpriteBatch, amin a rajzolás történik.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.

## SpikesTrap.cs

Egy tüskés csapda reprezentációjáért felelős.

**Metódusok:**

* Item(): A konstruktor beállítja az attribútumokat. Beállítja a collision boxot és az animációt.
  + Vector2 pos: Az csapda pozíciója.
* LoadContent(): Betölti a csapda fájljait.
  + ContentManager content: ContentManager amin keresztül a betöltés történik.
* Update(): Frissíti az csapda animációját. Ellenőrzi, hogy a karakter hozzáér-e az csapdához. Ha a karakter hozzáér az csapdához akkor megsebzi a karaktert.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.
* Draw(): Az csapda kirajzolását végzi el.
  + SpriteBatch spriteBatch: A SpriteBatch, amin a rajzolás történik.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.

# Maps

A Maps névérbe tartoznak a térkép betöltéséért, valamint a játék futásáért felelős osztályok.

## MapType.cs

A térképtípusokat felsoroló enum.

## Map.cs

Absztrakt osztály, ebből származnak le a térképek.

## BasicMap.cs

A BasicMap-et reprezentáló osztály. A Mapből származik le.

**Metódusok:**

* BasicMap(): A konstruktor beállítja az attribútumokat. Betölti a tértképhez szükséges fájlokat.
  + ContentManager content: ContentManager amin keresztül a betöltés történik.
  + GraphicsDevice graphicsDevice: GraphicsDevice, amivel a tértkép ki lesz rajzolva.

## Map2.cs

A Basic2-őt reprezentáló osztály. A Mapből származik le.

**Metódusok:**

* Map2(): A konstruktor beállítja az attribútumokat. Betölti a tértképhez szükséges fájlokat.
  + ContentManager content: ContentManager amin keresztül a betöltés történik.
  + GraphicsDevice graphicsDevice: GraphicsDevice, amivel a tértkép ki lesz rajzolva.

## MapManager.cs

A térkép betöltését, valamint a játék futtatását végző osztály.

**Metódusok:**

* MapManager(): A konstruktor beállítja az attribútumokat.
  + GameScene gameScene: A scene, amihez a manager tartozik
  + MapType mapType: A betöltendő térkép típusa.
  + PlayerChacterType playerType: A betöltendő jatékos típusa.
  + Game game: A futó Game1 példánya.
  + ContentManager content: ContentManager amin keresztül a betöltés történik.
* LoadObjects(): Betölti a térképet és az objektumokat.
* GenerateCollisionBox(): Collision boxot generál a betöltött térképelemhez.
* Update(): Frissíti az játék állapotát.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.
* Draw(): Frissíti az játék aktuális állapotát.
  + SpriteBatch spriteBatch: A SpriteBatch, amin a rajzolás történik.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.
* DrawBackground(): Frissíti a háttérképet.
* DrawMap(): Frissíti a térképet.
* CharacterDied(): Meghívódik ha egy karakter meghalt. Ha az összes karakter halott, akkor elindítja a játék végét.
* DeleteItem(): Töröl egy tárgyat a térképről.

# Scenes

A Scenes névtér a különböző képernyők kezeléséért felelős.

## Scene.cs

Absztrakt osztály. Ebből származnak le a sceenek.

**Metódusok:**

* Scene(): Absztrakt függvény. A scene frissítését végzi el.
* LoadContent(): Absztrakt függvény. A scene betöltését végzi el.
  + Game1 game: A Game amihez a scene tartozik.
  + ContentManager content: ContentManager amin keresztül a betöltés történik.
* Update(): Absztrakt függvény. A scene frissítését végzi el.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.
* Draw(): Absztrakt függvény. A scene kirajzolását végzi el.
  + SpriteBatch spriteBatch: A SpriteBatch, amin a rajzolás történik.
  + GameTime gameTime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.

## MenuScene.cs

A menüt reprezentáló scene. A Sceneből származik le.

## CharacterSelectorScene.cs

A karakterválasztót reprezentáló scene. A Sceneből származik le.

## MapSelectorScene.cs

A térképválasztót reprezentáló scene. A Sceneből származik le.

## GameScene.cs

A játékot reprezentáló scene. A Sceneből származik le.

## GameOverScene.cs

A játék végét reprezentáló scene. A Sceneből származik le.

# Game1.cs

A Game.cs osztály a játék futtatásáért felelős. A MonoGame Game osztályából származik le. Eltárolja az aktuálisan futó scenet és annak meghívja az Update, valamint Draw metódusait. Ezen az osztályon keresztül történik a különböző scene-ek közötti váltás.

**Metódusok:**

* Game1(): A konstruktor a játék alap paramétereinek beállításáért felelős.
* Initialize(): A különböző változók inicializálását végzi el.
* LoadContent(): A kezdőképernyő betöltéséért felelős.
* Update(): Minden játékciklusban meghívódik. Meghívja az aktuálisan aktív scene Update metódusát.
  + GameTime gametime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.
* Draw(): Minden játékciklusban meghívódik. Letörli a képernyőt, majd pedig meghívja az aktuálisan aktív scene Draw metódusát.
  + GameTime gametime: Ez a paraméter tartalmazza az előző hívás óta eltelt időt és a játék kezdete óta eltelt időt.
* ChangeScene(): A scene váltását végzi el.
  + Scene scene: Ez a paraméter lesz az új scene.