
ODIN'S
FURY

Tartalomjegyzék

Bevezető:

Tagok: Zovát Péter, Csiki Dávid, Kerekes Szabolcs

Tematika

A projektünk során egy középkori tematikájú videojáték készítése mellett döntöttünk, amely a viking és lovagi korszakok eseményeit ötvözi. Ennek az időszaknak a gazdag történelmi és kulturális háttere inspirálta a csapat minden tagját, így ez a téma kiváló alapot nyújtott egy izgalmas és vizuálisan lenyűgöző játék létrehozásához.

Játék Koncepció

Játék Motor

A játékunk fejlesztéséhez a Unity motorját választottuk, mivel ez a platform kiváló kompromisszumot nyújt a grafikai teljesítmény és a fejlesztési kényelem között. Az Unreal Engine, bár vizuálisan lenyűgözőbb eredményeket kínál, nagyobb hardverigényű, míg a Godot Engine programozási nyelvének keveréke számunkra kevésbé volt ideális, mivel nem rendelkeztünk elegendő idővel és erőforrással egy teljesen új nyelv elsajátítására. A Unity platformot C# programozási nyelven használtuk, amelyet már korábban is tanulmányoztunk, ezért ez a legpraktikusabb választásnak bizonyult.

Játék Koncepció

A játékunk egy összetett felhasználói felülettel rendelkezik, amely regisztráció és bejelentkezés után lehetővé teszi a játék letöltését. A belépés után a játékosok választhatnak a történeti mód és a földi csaták között. A történeti módban egy lore-t, azaz a játék történetét bemutató bevezetőt követően hajócsaták során vehetnek részt, ahol a felhasználók a tengeri stratégiák és taktikák különféle formáit sajátíthatják el. A negyedik csata után a játék a fő szakaszába lép, ahol a cél a király meggyilkolása különböző erősítések felhasználásával.

A játék karakter- és mozgásdesignja a Dark Souls és az Arkham Knight játékokból merített inspirációt. A karakterirányítás intuitív, az egérrel történő jobbra és balra

fordulás, valamint a bal egérgombbal történő ütés lehetőségével. A játék környezeti elemei, mint például a király trónja, a Game of Thrones sorozatból származnak, míg a király karaktere a The Witcher sorozatból kapott inspirációt.

UnityCloud

A projektmenedzsmentet és a csapatmunkát a Unity Cloud segítette, amely lehetővé tette a projekt távoli elérését és az egymástól független munkavégzést. Ez a megoldás nagyban hozzájárult a hatékony együttműködéshez és a projekt sikeres megvalósításához. A Unity-hez elérhető számos oktatóanyag és eszközkészlet tovább segítette a fejlesztést, lehetővé téve számunkra, hogy maximálisan kihasználjuk a rendelkezésre álló erőforrásokat és tudásanyagot.

Történet/ Lore

A történetünk, Sir Rodrick élete és küzdelmei köré épül, aki a középkori Skandinávia viharos időszakában, körülbelül 1000 körül, a mai Svédország területén él. Sir Rodrick, más néven a "királyölő", már fiatal korában, apja hatására, fegyverforgatásban és lovaglásban edződött, és már gyermekkorában részt vett csatákban és lovagi tornákon. Felnőttként már önállóan vezetett hadjáratokat az északi népek ellen, amelyek során veretlen maradt, egészen addig, amíg nem lépték át a mai Svédország határait.Egy ködös napon, Bjørn területén, egy erdővel övezett mezőn csaptak le Rodrick és seregeire. A túlerőben lévő ellenség legyőzte Rodrick csapatait, és Bjørn, saját felsőbbrendűségét hirdetve, a hősünket gladiátorként kényszerítette harcolni. Rodrick öt éven át minden második nap megvívott egy csatát, és mindegyiket megnyerte.

Eközben a király, egy önző, hataloméhes emberként, még a természetet is próbálta uralma alá hajtani. Parancsot adott a verebek kiirtására, nem gondolva arra, hogy ez milyen ökológiai következményekkel jár. A verebek hiánya miatt elszaporodtak a sáskák, amik elpusztították a terményt, ami éhínséget okozott a városban. A király bérkatonái leverték a forradalmat, amelyben a város lakói, nők és gyerekek is részt vettek.A zűrzavar alatt Sir Rodrick megszökött a fogságból, és elhatározta, hogy bosszút áll. Több évnyi barangolás után visszatért szülőföldjére, és első teendője volt hajóra szállni, hogy visszatérjen Svédországba. Útja során találkozott az angol hadihajókkal,

akik nem adták könnyen magukat. Az elbeszélés ezen pontján Rodrick története csúcsosodik ki, amint küzdelmei újabb szintre lépnek, és a történelmi Skandinávia sötét, viharos vizein navigál.Ez a történet kiváló alapot nyújt egy izgalmas, történelmi hátterű videojátékhoz, amely mélyreható karakterfejlődést és gazdag, részletesen kidolgozott világot kínál a játékosok számára. Innentől kezdődik a maga a Játék.

Az első scene:

az első scene-ami bemutatja a játék nevét és a mi „stúdiónkat” egy egyszerű fade in fade out scriptek történik a lényege az hogy a canvasoknak a canvas groupjának alap helyzete 0 és szépen fokozatosan emelkedik még élnem éri az egyet majd yield fieldel 10 secundum után fade outol majd jön a következő canvas :

```
float elapsedTime = 0f; // nullázás
float currentAlpha = canvasGroup.alpha; // átlátszóság beállítása

// Fade in ciklus
while (elapsedTime < fadeDuration)
{
   float newAlpha = Mathf.Lerp(currentAlpha, 1f, elapsedTime / fadeDuration); // Új átlátszóság
   canvasGroup.alpha = newAlpha; // CanvasGroup átlátszóságának beállítása
   elapsedTime += Time.deltaTime; // Eltelt idő frissítése
   yield return null; // Következő frame-re várakozás
}

canvasGroup.alpha = 1f; // Végső átlátszóság beállítás
```

Login/regisztráció: Regisztrációnál és a loginnál sql táblát használtunk, a regisztációhoz/ loginhoz a játékban PHP fájlokat használtunk ez úgy működik hogy a phpban a táblából lekérjük a felhasználói adatokat “$sql = "SELECT password FROM unitybackendtutorial WHERE username = '$username'"; “ majd a c#-on belül az inputfieldel beolvassuk a szöveget

```
LoginButton.onClick.AddListener(() =>
{
   StartCoroutine(main.Instance.Web.Login(UsernameInput.text,  PasswordInput.text));
  ok = UsernameInput.text;
});
```

majd ezt ahogy a változót áttesszük maga a web c#- ba és ott http requestel a PHP fájlból ami felvan töltve egy hostoló webre és és onnan lekéri ahogy az előző előtti kódban is látható és az vissza küldeni a c# kódnak és ha helyes a bejelentkezés akkor bedob a játék következő scenebe. A regisztráció is hasonló elveken alapszik azon felül hogy a regiszteruser.php-ban ugye nem lekérést alkalmazunk hanem insertet

$sql2 = "INSERT INTO unitybackendtutorial (username, password, level, coins)
VALUES ('$loginUser', '$loginPass', 0,.

A regisztrációnál is hasonló inputfieldel alkalmazunk:

* Regis.onClick.AddListener(() =>

{StartCoroutine(main.Instance.Web.RegisterUser(UsernameInput.text,
PasswordInput.text));

}); *

, a registeruser PHP-t meghívjuk ugyan így mint a login meghívásnál*
using (UnityWebRequest www =UnityWebRequest.Post
("https://mukodj123.000webhostapp.com/RegisterUser.php", form)) *

a coinquery és a coinupdate az az animalspawner scriptben belül alkalmazzuk, amint megöltünk egy wavet azután kapunk 200 coint ezt úgy érjük el hogy minden egyes alkalommal lekérdezzük phpn keresztül a coinokak *

```
public IEnumerator coinquery(string username)
{
   WWWForm form = new WWWForm();
   form.AddField("loginUser", username);  // Felhasználónév hozzáadása a formhoz

   using (UnityWebRequest www =
UnityWebRequest.Post("https://mukodj123.000webhostapp.com/coinquery.php",
form))
   {
      yield return www.SendWebRequest();  // Kérés küldése és várakozás a válaszra

      if (www.result != UnityWebRequest.Result.Success)  // Ellenőrizzük, hogy a kérés
sikeres volt-e
      {
         Debug.LogError(www.error);  // Ha nem, logoljuk az első hibát
         Debug.LogError(www.error);  // Duplikált hiba logolása - ez valószínűleg hiba,
elég egyszer logolni
         guestuser = true;  // Ha hiba történt, a felhasználót vendégként kezeljük

      }
      else
      {
         string responseText = www.downloadHandler.text;  // Válasz szövegének
kiolvasása
         if (int.TryParse(responseText, out int updatedCoins))            {
            coins = updatedCoins;  // Érmék számának frissítése
            Debug.Log("Coins updated to: " + coins);  // Logoljuk az új érmék számát
         }
         else
         {
```

```
            Debug.LogError("Failed to parse coins value from the response: " +
responseText);  // Ha nem sikerül értelmezni, logoljuk a hibát
        }
      }
   }
}
* és a coin kéréssel külön de megis egy idővel updateljük az adott coin menyiséget ami
esetünkben 200*
public IEnumerator coinupdate(string username, int coins)
{
   WWWForm form = new WWWForm();
   form.AddField("loginUser", username);  // Felhasználónév hozzáadása a formhoz
   form.AddField("coins", coins);  // Érmék számának hozzáadása a formhoz

   using (UnityWebRequest www =
UnityWebRequest.Post("https://mukodj123.000webhostapp.com/coinupdate.php",
form))
   {
      yield return www.SendWebRequest();  // Kérés küldése és várakozás a válaszra

      if (www.result != UnityWebRequest.Result.Success)  // Ellenőrizzük, hogy a kérés
sikeres volt-e
      {
         Debug.LogError(www.error);  // Ha nem, logoljuk a hibát
         guestuser = true;  // Hiba esetén a felhasználót vendégként kezeljük
      }
      else
      {
         Debug.Log("Coins updated successfully");  // Ha a kérés sikeres, logoljuk a
sikerességet
      }
   }
} *
```

A Történet elmesélése:

Mindenki szereti a Star Wars-t És arra gondoltunk, hogy az ikonikus Star Wars elején is látható szöveggel oldjuk meg a történet elmesélését. Először képekkel akartuk megoldani hasonló módon mint a bevezetőnél egy Fade In, Fade Out –al megoldani, de úgy ítéltük meg ,hogy ez sokkal igényesebb és egy szép utalás a Star Warsra. Maga a megvalósítás az animatorral történt, behúztunk egy canvast és belemásoltuk a szöveget és beillesztettük a Unity animátorába és a pozícióján lineárisan változtatunk, majd elhelyezzük a kamerát, olyan pozícióba, hogy hozza a Star Warsos hangulatot

Játék szerkezete:

Hajó scene: a víz fekete mivel ez szó szerinti fekete szoros. A vízben a scriptben megírt módon hullámzik *

```
void Update()
{
   Vector3[] vertics = meshFilter.mesh.vertices; // Az objektum mesh-ének csúcspontjai (vertices)
   for (int i = 0; i < vertics.Length; i++) // Iterálás minden csúcspontra
   {
      // Beállítja a csúcs Y koordinátáját a hullám magasságának megfelelően
      vertics[i].y = wavemeneger.instance.GetWaveHeight(transform.position.x +
         vertics[i].x);
   }
   meshFilter.mesh.vertices = vertics; // Frissíti a mesh csúcsait az új pozíciókkal
   meshFilter.mesh.RecalculateNormals(); // Újraszámolja a normálvektorokat a megvilágítás helyességének biztosítása érdekében
}
```

* Ez azért is fontos mivel a hajó mozgásában jobbra balra illetve előre hátra dölöngél a víz hullámzásával együtt ezt 4 floaterrel értük el így módon*

```
public Rigidbody rigidbody; // A test, amire a fizikai hatások alkalmazódnak
public float depthbeforesubmerged = 1f; // A merülési mélység előtti beállítás
public float displacementamount = 3f; // A felhajtóerő módosításának mennyisége
public int floatercount = 1; // A lebegő testek száma
public float waterDrag = 0.5f; // A vízi ellenállás beállítása (alacsonyabb érték nagyobb ellenállás)
public float waterAngularDrag = 0.5f; // A vízi forgási ellenállás beállítása

private void FixedUpdate()
{
   // Az alapvető gravitációs erő hozzáadása a testhez, felosztva a lebegők számával
   rigidbody.AddForceAtPosition(Physics.gravity / floatercount, transform.position, ForceMode.Acceleration);

   float waveHeight = wavemeneger.instance.GetWaveHeight(transform.position.x); // A hullám magasságának lekérése

   if (transform.position.y < 0f) // Ellenőrzi, hogy a test víz alatt van-e
   {
      // A felhajtóerő kiszámítása, ami arányos a víz alatti mélységgel
      float displacementmultiplier = Mathf.Clamp01((waveHeight - transform.position.y) / depthbeforesubmerged) * displacementamount;

      // Felhajtóerő hozzáadása a testhez
      rigidbody.AddForceAtPosition(new Vector3(0f, Mathf.Abs(Physics.gravity.y) * displacementmultiplier, 0f), transform.position, ForceMode.Acceleration);

      // Vízi ellenállás hozzáadása a sebesség csökkentése érdekében
      rigidbody.AddForce(displacementmultiplier * -rigidbody.velocity * waterDrag * Time.fixedDeltaTime, ForceMode.VelocityChange);

      // Vízi forgási ellenállás hozzáadása a forgó mozgás csökkentése érdekében
```

```
        rigidbody.AddTorque(displacementmultiplier * -rigidbody.angularVelocity *
waterAngularDrag * Time.fixedDeltaTime, ForceMode.VelocityChange);
    }
}
*
```

az irányítás a D és az A gombokkal lehet rotálni a hajót, a hajó a vector 3on mindig előre megy ahogy a többi hajó is • Az ágyú az az E és a Q betűkkel lehet lőni Q betűvel balra E betűvel jobbra a lövés így történik *

```
public List<Transform> bulletSpawnPoints; // A bullet pontok listája
public GameObject bulletPrefab; // A bullet prefab, amit a játék során használnak
public float bulletSpeed = 10; // A bulletsebessége
public float cooldownTime = 5f; // Hűtési idő másodpercben
private bool isOnCooldown = false; // Jelző, ami nyomon követi, hogy a fegyver hűtés
alatt áll-e

void Update()
{
    // Ellenőrzi, hogy a fegyver nincs-e hűtés alatt, és az 'E' billentyűt lenyomták-e
    if (!isOnCooldown && Input.GetKeyDown(KeyCode.E))
    {
        // Lőszer kilövés minden indítási pontból
        foreach (Transform spawnPoint in bulletSpawnPoints)
        {
            FireBullet(spawnPoint);
        }

        // Hűtési idő kezdetének indítása
        StartCoroutine(StartCooldown());
    }
}

void FireBullet(Transform spawnPoint)
{
    // bullet példányosítása a megadott helyen és orientációval
    var bullet = Instantiate(bulletPrefab, spawnPoint.position, spawnPoint.rotation);

    // A bullet sebességének beállítása
    bullet.GetComponent<Rigidbody>().velocity = spawnPoint.forward * bulletSpeed;
}

IEnumerator StartCooldown()
{
    // A fegyver állapotának beállítása hűtés alattire
    isOnCooldown = true;

    // Várakozás a hűtési idő leteléséig
    yield return new WaitForSeconds(cooldownTime);

// Hűtési idő lejárta, újra tüzelhet
    isOnCooldown = false;
```

```
}
 ha a lövedék eltalálja az adott taggel ellátott objektumot akkor az ellenféltől golyónként levesz 10 hp-t illetve destroyolja a gameobjectet *
public float life = 10; // A játékobjektum élettartama másodpercben

void Awake()
{
   Destroy(gameObject, life); // A játékobjektum automatikus megsemmisítése a meghatározott idő elteltével
}

void OnCollisionEnter(Collision collision)
{
   // Ütközésdetektálás: ellenőrizzük, hogy az ütközött objektum címkéje "enemyship"-e
   if (collision.gameObject.CompareTag("enemyship"))
   {
      Destroy(gameObject); // Az aktuális játékobjektum megsemmisítése az ütközés hatására
      boat.hp -= 10; // A "boat" objektum életerejének csökkentése 10 egységgel
   }
}
* Az ágyúgolyó kilövéséhez 8 spawnert használtunk négy baloldalt 4 jobboldalt: *

public List<Transform> bulletSpawnPoints; // A bullet pontok listája

* Ha az ellenséges hajó, vagy a mi hajónk nekimegy a hegynek 180 fokot fordul a hajó :

*

 else if (collision.gameObject.CompareTag("mountain"))
{
   // A hajó 180 fokos elforgatása felfelé mutató tengely (Y-tengely) körül
   transform.Rotate(Vector3.up, 180f);
}
* Az ellenséges hajók mozgása random módra történi adott random rotationon belül és időn belül kanyarodik a hajóval . Az ellenséges hajó ágyúkilövései aszerint vannak megoldva hogy mind a bal mind a jobboldalra raktunk egy empty game objektumot és ha azokkal collidol akkor elkezdik a lövést*
public GameObject aliship; // A célhajó objektum
public GameObject cannonballPrefab; // Az ágyúgolyó prefab
public List<Transform> cannonballSpawnPoints; // Az ágyúgolyó indítási pontjainak listája
public float cannonballSpeed = 10f; // Az ágyúgolyó sebessége
public float cooldownTime = 5f; // cooldown idő másodpercben

private bool isInRange = false; // Jelző, hogy a célhajó hatótávolságban van-e
private bool isOnCooldown = false; // Jelző, hogy az ágyú épp cooldown alatt áll-e

private void OnTriggerEnter(Collider other)
{
   if (other.gameObject == aliship) // Ellenőrzi, hogy a kapcsolatba lépő objektum a célhajó-e
   {
```

```
        isInRange = true; // Beállítja a hatótávolság jelzőt igazra
        Debug.Log("Targeted"); // Naplózza, hogy a célhajó hatótávolságban van
    }
}

private void OnTriggerExit(Collider other)
{if (other.gameObject == aliship) // Amikor a célhajó kilép a hatótávolságból
    {
        isInRange = false; // Beállítja a hatótávolság jelzőt hamisra
    }
}

private void Update()
{
    if (isInRange && !isOnCooldown) // Ha a célhajó hatótávolságban van és nincs hűtés
alatt
    {
        ShootCannonballs(); // Ágyúgolyók kilövése
    }
}

private void ShootCannonballs()
{
    foreach (Transform spawnPoint in cannonballSpawnPoints) // Minden ágyúgolyó
indítási pontnál
    {
        // Ágyúgolyó példányosítása az adott helyen és irányban
        GameObject cannonball = Instantiate(cannonballPrefab, spawnPoint.position,
spawnPoint.rotation);
        // Az ágyúgolyó sebességének beállítása
        cannonball.GetComponent<Rigidbody>().velocity = spawnPoint.forward *
cannonballSpeed;
    }
    // Beállítja a cooldown jelzőt igazra
    isOnCooldown = true;
    // Hűtési időt követően visszaállítja a cooldown
    Invoke("ResetCooldown", cooldownTime);
}

private void ResetCooldown()
{
    // A cooldown alatt álló jelző visszaállítása hamisra
    isOnCooldown = false;}
*
```

Szárazföldi játék:

A játék tervezete az alábbi forgató könyvön alapszik egy bejövetel hajóval, majd a mólónál megáll és a Colosseum felé elindulva és oda belépve elindulnak a katonák akikből 4 hullám jön majd azután tovább lehet haladni életerőt vásárolni varázslatot

venni ezen kívül akár Blackjackezni sőt ki mit tud kérdésekre válaszolni. Majd ha bevásároltunk bemehetünk a kastélyba ahol tank roleba beillő harcosokkal küzdünk meg egy nagy várban aminek a közepén a throneába ül a Király majd ha itt is kitakarítottuk a 4 darab hullámot a király felkell és különböző képességekkel dobálni, és ezzel vége is a játéknak. Hogy minél több coint szedj össze és fejleszt tovább a karaktereket játssz minél többet (hogy felkerülj a toplisták élére) és minél hamarabb minél könnyebben legyőzd a gonosz királyt A hajót itt nem te irányítod automatikusan megy amíg „shipcollider”-hez nem ütközik *

```
public Transform target; // Ez az érték beállítható az Inspector panelen
public float speed = 5f; // A hajó mozgásának sebessége
private bool move = true; // Zászló a mozgás szabályozására
public Transform player; // Hivatkozás a játékosra

void Start()
{
   // Választhatóan megkeresi a játékost címke vagy név alapján, ha ez nem lett beállítva az Inspectorban
   if (player == null)
   {
      player = GameObject.FindGameObjectWithTag("Player").transform;
   }
}

void Update()
{
   if (move && target != null)
   {
      // Kiszámítja a mozgáshoz szükséges távolságot ebben a képkockában
      float step = speed * Time.deltaTime;

      // Mozgatja a hajót a cél felé
      transform.position = Vector3.MoveTowards(transform.position, target.position, step);

      // Mozgatja a játékost ugyanazzal a sebességgel a cél felé
      if (player != null)
      {
         player.position = Vector3.MoveTowards(player.position, target.position, step);
      }
   }
}

private void OnCollisionEnter(Collision collision)
{
   // Ellenőrzi, hogy a hajó ütközik-e a céllal
   if (collision.gameObject == target.gameObject)
   {
```

```
        move = false; // Megállítja a hajó és a játékos mozgását
        Debug.Log("Ütközés a céllal! A hajó megállt.");
    }
}
```
* azután a foward speed 0 rá állítjuk A mólónál ügyelni kell arra hogy nehogy beleessünk a vízbe mert ha hozzáérünk a vízhez a karakterünk meghal

code
Miután leérkeztünk a dokkhoz egy irányba tudunk menni a collusseum felé amikor belépünk elindulnak a warriorok ezt úgy értük el hogy egy átlátszó collidert leraktunknami istriggered azaz átlehet menni rajta és aktiválja a playertriggeredt *

```
private Collider _collider; // Collider típusú privát változó a komponens tárolására

void Start()
{
    _collider = GetComponent<Collider>(); // A Collider komponens lekérdezése a játékobjektumról
    if (_collider == null)
    {
        // Hibalog, ha a szükséges Collider komponens hiányzik a játékobjektumról
        Debug.LogError("A szükséges Collider komponens hiányzik az objektumról.", this);
    }
    else
    {
        // Beállítja a Collider komponenst úgy, hogy az trigger legyen
        _collider.isTrigger = true;
    }
}

private void Update()
{
    // Feltételvizsgálat, hogy az állat aktiválta-e a triggert és nem haltak-e meg az összes állat
    if (Animal3.playerTriggered && !AnimalSpawner.diedall)
    {
        // Elindít egy Coroutine-t, ami átállítja a Collider-t szilárdra
        StartCoroutine(SetColliderSolid());
    }
    else
    {
        // Beállítja a Collider-t triggerre, ha nem teljesülnek a feltételek
        _collider.isTrigger = true;
    }
}

IEnumerator SetColliderSolid()
{
    yield return new WaitForSeconds(2); // Várakozás 2 másodpercet

    if (_collider != null)
    {
```

```
        // Ha a Collider még létezik, beállítja, hogy ne legyen trigger
        _collider.isTrigger = false;
        // Logolja, hogy a Collider most már szilárd
        Debug.Log("A Collider most már szilárd!", this);
    }
} *
```

A warriorok mozgása a következő működik :

```
// Ha a játékos aktiválta és a karakter még nem halott
if (playerTriggered && !dead)
{
    // Ha a karakter még nem halt meg
    if (!isdie)
    {
        // Mozgás a játékos felé
        MoveTowardsPlayer();
    }
}

public void MoveTowardsPlayer()
{
    // Ellenőrzi, hogy a cél (játékos) létezik-e
    if (player != null)
    {
        // Kiszámítja a távolságot a jelenlegi pozíció és a cél között
        float distance = Vector3.Distance(transform.position, target.position);

        // Ha a távolság kisebb vagy egyenlő a megállási távolsággal
        if (distance <= stoppingDistance)
        {
            // Támadási animáció lejátszása
            anim.CrossFade("attack");
            // Az AI útvonaltervező útvonalának törlése
            agent.ResetPath();
            // Animációs állapot beállítása igazra
            isanim = true;
        }
        else
        {
            // Beállítja a cél pozícióját az AI útvonaltervezőben
            agent.SetDestination(target.position);
            // Futó animáció lejátszása
            anim.CrossFade("run");
        }
    }
}
```

A bolt

megtalálható egy házban ahol p-t lehet venni ha elég közel vagy az árushoz akkor a talán P gomb megnyomásával bejön egy canvas hogy press K for gain HP és ha ezt

megtesszük akkor levon egy 10 coint hogy kapj 10 hp-t itt jön be az előző coin query és coinupdate olyan szempontból ha veszünk valamit akkor az adatbázisból lekéri a coinokat majd levonja:

```
public Button hp; // Referencia a HP gombra
public float activationDistance = 5f; // Aktiválási távolság küszöbe
private Player player; // Referencia a játékos objektumra
public AnimalSpawner animalSpawner; // Hivatkozás az állatok spawnolásáért felelős objektumra, állítsa be az Unity Editorban
private int coinsToSubtract = 10; // Levonandó érmék mennyisége
public GameObject menu; // Menü objektum
private bool menuenabled = false; // Menü engedélyezett állapotának jelzője
private Animation anim; // Animációs komponens
public float fadeDuration = 2.0f; // Halványítás időtartama
[SerializeField] private CanvasGroup canvasGroup; // Canvas csoport a UI elemeinek kezelésére

private void Start()
{
   // A játékos komponens lekérése a tag alapján
   player = GameObject.FindGameObjectWithTag("Player").GetComponent<Player>();
}

private void Update()
{
   // Ellenőrzi, hogy a játékos a meghatározott távolságon belül van-e
   if (Vector3.Distance(transform.position, player.transform.position) <= activationDistance)
   {
      StartCoroutine(SequentialFade()); // Szekvenciális halványítás indítása

      if (Input.GetKeyDown(KeyCode.K))
      {
         BuyHealth(); // Egészség vásárlása, ha a 'K' billentyűt lenyomják
      }
   }
   else
   {
      canvasGroup.alpha = 0; // Ha a játékos túl messze van, állítsa az átlátszóságot 0-ra
   }
}

private void BuyHealth()
{
   // Ellenőrzi, hogy elegendő érme áll-e rendelkezésre az egészség vásárlásához
   if (animalSpawner != null && animalSpawner.coins >= coinsToSubtract)
   {
      animalSpawner.coins -= coinsToSubtract; // Érmék levonása
      player.health += 10f; // Játékos egészségének növelése
      StartCoroutine(animalSpawner.coinupdate(animalSpawner.k, animalSpawner.coins)); // Érmék frissítése
```

```
        Debug.Log("Játékos egészsége: " + player.health);
        Debug.Log("Érmék: " + animalSpawner.coins);
    }
    else
    {
        Debug.Log("Nincs elég érme az egészség vásárlásához vagy az AnimalSpawner
referencia nem található.");
    }
}

public IEnumerator SequentialFade()
{
    // A jelenlegi CanvasGroup fokozatos beállítása
    yield return StartCoroutine(FadeIn(canvasGroup));
}
```

Quiz Game

A shop után meglátogathatjuk a ki mit tudos emberünket, akik kérdések feltételével tudunk coint szerezni és ha helyesen válaszolunk, akkor coint kapunk, ha a megfelelő távolságban állunk és megnyomjuk a P betűt akkor felugrik egy canvas ahol gombok segítségével tudunk válaszolni az adott kérdésekre a canvasok felugrása a következő módon történik

CODE:

```
private void Start()
{
    // A játékos komponens lekérdezése a 'Player' tag alapján
    player = GameObject.FindGameObjectWithTag("Player").GetComponent<Player>();
}

private void Update()
{
    // Ellenőrzi, hogy a játékos a meghatározott távolságon belül van-e
    if (Vector3.Distance(transform.position, player.transform.position) <=
activationDistance)
    {
        StartCoroutine(SequentialFade()); // Szekvenciális halványítás indítása
        if (Input.GetKeyDown(KeyCode.L)) // Ha az 'L' gombot lenyomják
        {
            if (!menuenabled) // Ha a menü még nincs engedélyezve
            {
                buyluck(); // Szerencse vásárlásának kísérlete mielőtt a menü engedélyezésre
kerül
            }
            else
            {
                menuenabled = false; // Menü kikapcsolása
```

```
            DisableCanvases(); // Vásznak letiltása
        }
    }

    if (menuenabled && coinsDecreased) // Ha a menü engedélyezve van és az érmék száma csökkent
    {
        Canvas randomCanvas = canvases[Random.Range(0, canvases.Count)]; // Véletlenszerű vászon kiválasztása
        randomCanvas.enabled = true; // Vászon engedélyezése
        open = true; // Jelezze, hogy nyitva van
    }
    else
    {
        open = false; // Jelezze, hogy zárva van
        DisableCanvases(); // Összes vászon letiltása
    }
  }
  else
  {
    canvasGroup.alpha = 0; // Ha a játékos túl messze van, átlátszóságot 0-ra állít
  }
}

private void buyluck()
{
  // Ellenőrzi, hogy van-e elegendő érme szerencse vásárlásához
  if (animalSpawner != null && animalSpawner.coins >= coinsToSubtract)
  {
    animalSpawner.coins -= coinsToSubtract; // Érmék levonása
    StartCoroutine(animalSpawner.coinupdate(animalSpawner.k, animalSpawner.coins)); // Érmék frissítése
    coinsDecreased = true; // Jelző beállítása igazra, miután az érmék csökkentve lettek
    menuenabled = true; // Menü engedélyezése, ha az érmék sikeresen csökkentve lettek
  }
  else
  {
    Debug.Log("Nincs elég érme a szerencse vásárlásához vagy az AnimalSpawner referencia nem található.");
    coinsDecreased = false; // Jelző beállítása hamisra
    menuenabled = false; // Menü letiltása
  }
}
```

randomizált canvasok 5 darab canvas közül dobál fel opciókat majd külömbötő

ajándékokat

CODE:

kapunk(+ hp , +attack , coin):

```
public Button bt1; // Referencia az első gombra
public Button bt2; // Referencia a második gombra
public Button bt3; // Referencia a harmadik gombra
private Player player; // Referencia a játékosra
public GameObject menu; // Referencia a menüre

void Start()
{
   // A játékos komponens lekérdezése a 'Player' tag alapján
   player = GameObject.FindGameObjectWithTag("Player").GetComponent<Player>();

   // Eseményhallgatók hozzáadása a gombokhoz
   bt1.onClick.AddListener(() => OnButtonClick(bt1, false)); // Hamis válasz gomb
   bt2.onClick.AddListener(() => OnButtonClick(bt2, true)); // Helyes válasz gomb
   bt3.onClick.AddListener(() => OnButtonClick(bt3, false)); // Hamis válasz gomb
}

void OnButtonClick(Button clickedButton, bool correct)
{
   // Ha a helyes gombot nyomták meg
   if (correct)
   {
      player.health += 30f; // Növeli a játékos életét
      Debug.Log("jó2"); // Kiírja, hogy jó válasz
   }
   else
   {
      player.health -= 10; // Csökkenti a játékos életét
   }

   // Letiltja a gombok interakcióját
   SetButtonsInteractable(false);
}

void SetButtonsInteractable(bool interactable)
{
   // Beállítja a gombok interaktívképességét
   bt1.interactable = interactable;
   bt2.interactable = interactable;
   bt3.interactable = interactable;
}

public void EnableButtons()
{
   // Újra engedélyezi a gombok interakcióját
   SetButtonsInteractable(true);
}
```

MAGE:

Ha megválaszoltuk a kérdéseket akkor mehetünk tovább a magehez akit szintén meg kell keresnünk ha akarunk tűzgolyót venni, a tűzgolyó egy hasonló scriptel van ellátva mint a hajó ágyúgolyója

```
// Nyilvános változó, amely meghatározza az objektum élettartamát másodpercben.
public float life = 3;

// Privát változó a játékos objektum tárolására.
private Player player;

// Amikor az objektum először létrejön, automatikusan meghívódik ez a függvény.
void Awake()
{
    // Megsemmisíti az objektumot a megadott élettartam után.
    Destroy(gameObject, life);
}

// A játék kezdete előtt hívódik meg ez a függvény.
private void Start()
{
    // Megkeresi a játékost azonosító tag alapján, és hozzárendeli a 'player' változóhoz.
    player = GameObject.FindGameObjectWithTag("Player").GetComponent<Player>();
}

// Ez a függvény akkor hívódik meg, amikor az objektum ütközik egy másik objektummal.
void OnCollisionEnter(Collision collision)
{
    // Ellenőrzi, hogy az ütközött objektum címkeje "Animal" e.
    if (collision.gameObject.CompareTag("Animal"))
    {
        // Megsemmisíti az állat objektumot.
        Destroy(collision.gameObject);
        // Megsemmisíti ezt az objektumot is.
        Destroy(gameObject);
    }
    // Ellenőrzi, hogy az ütközött objektum címkeje "Player" e.
    else if (collision.gameObject.CompareTag("Player"))
    {
        // Csökkenti a játékos egészségét 10 egységgel.
        player.health -= 10;
        // Megsemmisíti ezt az objektumot.
        Destroy(gameObject);
    }
}
```

van rajta egy kürülbelül 25 seces számlálóval van felruházva ami azt csinálja hogy nem tudsz tűzgolyókat lőni csak 25 másodpercenként azaz meg kell válogatnod mire használod

Tovább menve a kastélyba találkozunk a tank enemykkel akiknek van még 100 plusz armorja azaz másfélszer nehezebb őket megölni az ellenfelek mozgása és animációja az animal scriptben történik ahol unityengine.ai segítségével a navmesh surficen mozognak a player felé

```
// A játékos felé mozdulásért felelős metódus.
public void MoveTowardsPlayer()
{
   // Ellenőrzi, hogy a 'player' változó nem üres-e.
   if (player != null)
   {
      // Kiszámítja a távolságot a jelenlegi pozícióból a célpont pozíciójáig.
      float distance = Vector3.Distance(transform.position, target.position);

      // Ha a távolság kisebb vagy egyenlő a megállási távolsággal.
      if (distance <= stoppingDistance)
      {
         // Támadó animáció lejátszása.
         anim.CrossFade("attack");

         // Az útvonaltervező (agent) útvonalának törlése.
         agent.ResetPath();

         // Animáció állapotjelző beállítása igazra.
         isanim = true;
      }
      else
      {
         // Az útvonaltervező céljának beállítása a játékos pozíciójára.
         agent.SetDestination(target.position);

         // Futó animáció lejátszása.
         anim.CrossFade("run");
      }
   }
}

// Korutin, amely megsemmisíti az objektumot a megadott késleltetés után.
IEnumerator DestroyAfterDelay(float delay)
{
   // Várakozik a megadott késleltetési időre.
   yield return new WaitForSeconds(delay);

   // Megsemmisíti ezt a GameObject-et.
   Destroy(gameObject);
}

// A játékos felé fordulásért felelős metódus.
public void turnTowardsPlayer()
{
```

```
    // Ellenőrzi, hogy a 'player' változó nem üres-e.
    if (player != null)
    {
        // Hibaüzenet kiírása a konzolra a fordulásról.
        Debug.Log("Turning towards player");

        // Kiszámítja az irányvektort a játékos felé.
        Vector3 directionToPlayer = player.transform.position - transform.position;

        // Az irányvektor y komponensének nullázása (hogy ne forduljon felfelé vagy
lefelé).
        directionToPlayer.y = 0;

        // A megfelelő forgás kiszámítása a játékos felé.
        Quaternion lookRotation = Quaternion.LookRotation(directionToPlayer);

        // Sima forgatás az aktuális forgásból a kiszámított forgásba.
        transform.rotation = Quaternion.Slerp(transform.rotation, lookRotation,
Time.deltaTime * rotationSpeed);
    }
    else
    {
        // Hibaüzenet kiírása a konzolra, ha a játékos nem található.
        Debug.Log("Player not found");
    }
}
```

annak érdekében hogy ne másszanak a player arcában megy 2.7es távolságot tartanak mielőtt aktiválódik az attack animáció. Az animációkat nem animator segítségével hanem a scriptben a Crossfade segítségével oldottuk meg Ami úgy működik hogy létrehotunk egy animation változót amit esetünkben elneveztünk anim-nak éd ha az adott feltétel teljesül és így az animáció átvált és ha arréb megyünk akkor ugyan úgy a run animació indul el. Az attack methodot úgy oldottuk meg ha az ellenfél fejszéje/kardja eltalál minket collideol velünk akkor a hpnkat leveszi

```
// Játékos
private Player player;
// érintkezett-e az objektummal.
private bool hasHitPlayer = false;
 // automatikusan meghívódó függvény.
private void Start()
{
    // Megkeresi a játékost .
    player = GameObject.FindGameObjectWithTag("Player").GetComponent<Player>();
}

// Collider belép egy másik Colliderbe.
public void OnTriggerEnter(Collider other)
{
```

```
    if (other.tag == "Player" && !hasHitPlayer)
    {
        // Csökkenti a játékos életét 10-el
        player.health -= 10;

        // Beállítja igazra.
        hasHitPlayer = true;

        // Megkezdi az ütés állapot resetét
        StartCoroutine(ResetHitStatus());
    }
}

// 2 másodperc várakozás után hitel.
IEnumerator ResetHitStatus()
{
    // Várakozik 2 másodpercet.
    yield return new WaitForSeconds(2f);

    // Beállítja hamisra.
    hasHitPlayer = false;
}
```

itt azt is megkellet oldanunk hogy mivel update-ba írtuk így miközben collide-ol aközben többször veszi le a hp-nkat magyarán hiába csak egyszer üt meg az ellenfél sokkal több hp-t folyamatosan levesz emiatt ezt a így tudtuk megoldani hogy ez rendesen müködjön és ne bugoljon ilyen módódon

Íjász: Az íjász

Az íjász a vár falán állnak a trükkje az ha a trigerrel collidol (Ahogy a Player scriptbe már bemutattam),akkor elkezd forogni a és ha látja a játékost akkor elkezd lőni. // A játékos felé fordulás

```
public void turnTowardsPlayer()
{
    // Ellenőrzi, hogy a 'player' változó létezik
    if (player != null)
    {
        // Kiszámítja az vektort a játékos felé.
        Vector3 directionToPlayer = player.transform.position - transform.position;

        // forgás csak a vízszintes síkban történjen
        directionToPlayer.y = 0;

        // A megfelelő forgás
```

```
        Quaternion lookRotation = Quaternion.LookRotation(directionToPlayer);


        transform.rotation = Quaternion.Slerp(transform.rotation, lookRotation,
Time.deltaTime * rotationSpeed);
    }
    else
    {

    }
}

void Update()
{

    if (!isOnCooldown && arrowneni.playerTriggered)
    {
        // Lő egy lövedéket a FireBullet hívásával
        FireBullet();
        // Elindítja a cooldown-t
        StartCoroutine(StartCooldown());
    }

}

// A lövedék kilövése
void FireBullet()
{
    // golyó lespawnoltatása
    var bullet = Instantiate(bulletPrefab, bulletSpawnPoint.position,
bulletSpawnPoint.rotation);

    // Beállítja a lövedék sebességét a kilövési pont előre irányába
    bullet.GetComponent<Rigidbody>().velocity = bulletSpawnPoint.forward *
bulletSpeed;
}
```

A nyílak mint a player magicnél vagy, mint az ágyúgolyóknál múködik scak itt van rajta egy animáció.

A KIRÁLY/BOSS

A BOSS tűzgolyókat lő ki a player irányában hasonló scriptben mint az íjász annyi különbséggel hogy van egy stun, mikor eltelik 10 secundum akkor legugol és addig lehet ütni majd 6 sec múlva kezdi előröl

```
IEnumerator Stunned()
{

    stunned = true;
```

```
    // animáció lejátszása.
    anim.CrossFade("knee");

    // Átállítja a 'stunner123' változót hamisra, ami jelzi, hogy a karakter stunolt
    stunner123 = false;

    // Várakozás 4 másodpercet.
    yield return new WaitForSeconds(4);

    // Térdelésből felállás animáció
    anim.CrossFade("kneeup");

    // Várakozás 2 másodpercet.
    yield return new WaitForSeconds(2);

    // Az időmérő változó visszaállítása 0
    time1 = 0; // Visszaállítja az időt a bénulás után

    // Átállítja a 'stunner123' változót igazra
    stunner123 = true;

   // stun állapotot hamisra.
    stunned = false;
}
```

Itt tároljuk a levelt és adunk hozzá egyet amivel felkerülhetsz a leaderboardra a weboldalon

```
    public IEnumerator levelupdate(string username, int level)
    {
        // Új form létrehozása adatok postolásához.
        WWWForm form = new WWWForm();
        // A felhasználónév hozzáadása a formhoz.
        form.AddField("loginUser", username);
        // A szint hozzáadása a formhoz.
        form.AddField("level", level);

        // UnityWebRequest használata a szerverhez való postoláshoz.
        using (UnityWebRequest www =
UnityWebRequest.Post("https://mukodj123.000webhostapp.com/levelupdate.php",
form))
        {
            // Várakozás a válaszra a webszerverről.
            yield return www.SendWebRequest();

            // Hibakezelés, ha a kérés nem sikerült.
            if (www.result != UnityWebRequest.Result.Success)
            {
```

```
            Debug.LogError(www.error);
        }
        else
        {
            // Sikeres frissítés logolása.
            Debug.Log("level updated successfully");
        }
    }
}

// Egy korutin, amely lekérdezi a felhasználó szintjét.
public IEnumerator levelquery(string username)
{
    // Új form létrehozása adatok postolásához.
    WWWForm form = new WWWForm();
    // A felhasználónév hozzáadása a formhoz.
    form.AddField("loginUser", username);

    // UnityWebRequest használata a szerverhez való postoláshoz.
    using (UnityWebRequest www =
UnityWebRequest.Post("https://mukodj123.000webhostapp.com/levelquery.php",
form))
    {
        // Várakozás a válaszra a webszerverről.
        yield return www.SendWebRequest();

        // Hibakezelés, ha a kérés nem sikerült.
        if (www.result != UnityWebRequest.Result.Success)
        {
            Debug.LogError(www.error);
        }
        else
        {
            // A válasz szövegének kiolvasása.
            string responseText = www.downloadHandler.text;
            // A válaszból kiolvasott érték konvertálása egész számmá és hozzárendelése a
level változóhoz.
            if (int.TryParse(responseText, out int updatedCoins))
            {
                level = updatedCoins;
                Debug.Log("Coins updated to: " + level);
            }
            else
            {
                // Hiba logolása, ha a válasz szövegéből nem sikerül kiolvasni az értéket.
                Debug.LogError("Failed to parse coins value from the response: " +
responseText);
            }
        }
    }
}
```

Ha megöltük a királyt a die animation után beloadingol a Credits screenre ahol láthatjuk azt hogy ki milyen munkafolyamatokba vett részt a játékban.

Az animációkat mixamoról szedtük le ez egy ingyenes animátor weboldal ahová feltöltünk egy objektumot és kikeressük azt az adott animációt amit keresünk kitudjuk választani és onnan könnyedén letölteni ennek nagy előnye hogy időt spórolunk illetve hogy az animációhoz hozzátudjuk tenni a gameobjectekben a mi esetünkbe a kardot vagy a fejszét és vele eggyüt mozog és nem kell a küldönböző pozíciókkal és mozgásokkal törődni. • A bevezetőnél az insprirációnál már említettem a karakter mozgást, az egérrel tudunk fordulni hasonlóan mint egy FPS játékban vagy a Batman Arkham seriákban. Sokat gondolkodtunk hogy így oldjuk meg a mozgást vagy rendes wasd mozgással de végül arra a döntésre jutottunk hogy az animáció kissebb kihívásai miatt, illetve az irányítás is egy fokkal egyszerűbb. Az előre hátra oldalra mozgást a wasd vél oldottuk meg az animációkat egyszerűen ha az Inputkey w akkor a run animaciót játsza ha az a-t vagy a d-t akkor a side animációt ha az s-t akkor a backwards animációt A lovag ütését triggeringAi-al oldottuk meg az a lényeg, ha egy adott távolságon belül van a a képernyőn az ellenfél és balegér inputot adunk.

Weboldal

Regisztráció: Itt tudunk regisztrálni egyszerű elve van egy insertel a táblába feltöljük a felhasználót ahonnan a loginba, és a játékban is letudjuk kérni és bejelentkezni.

```
include 'db.php';
// Post method használása
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $conn->real_escape_string($_POST['username']);
    $password = $conn->real_escape_string($_POST['password']);

    // Megnézzük hogy létezik-e
    $result = $conn->query("SELECT * FROM unitybackendtutorial WHERE username
= '$username'");

    if ($result->num_rows > 0) {
// ha létezik már ilyen felhasználó
        echo "<div class='alert alert-danger' role='alert'>Error: Username already
exists.</div>";
    } else {
// Insert into-val beillesztjük a megfelelő oszlopba és sorba
        $stmt = $conn->prepare("INSERT INTO unitybackendtutorial (username,
password, level, coins) VALUES (?, ?, ?, ?)");
        $stmt->bind_param("ssii", $username, $password, $level, $coins);
```

```
// A szint és a coin kezdő értéke 0
        $level = isset($_POST['level']) ? $_POST['level'] : 0;
        $coins = isset($_POST['coins']) ? $_POST['coins'] : 0;

        if ($stmt->execute()) {
//ha sikeres
            echo "<div class='alert alert-success' role='alert'>New record created
successfully</div>";
            echo "<p><a href='login1.php' class='btn btn-primary'>Go to Login</a></p>";
        } else {
            echo "<div class='alert alert-danger' role='alert'>Error: " . $stmt->error .
"</div>";
        }
        $stmt->close();
    }

    $result->free();
    $conn->close();
}
```

Login: A táblából lekéri az adatokat és megnézi hogy egyeznek-e az inputfildes adatokkal

```
session_start();
include 'db.php';

$error_message = ""; // error message

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $conn->real_escape_string($_POST['username']);
    $password = $conn->real_escape_string($_POST['password']);
    //lekérés az adatbázisból
    $sql = "SELECT password FROM unitybackendtutorial WHERE username =
'$username'";
    $result = $conn->query($sql);

    if ($result->num_rows == 1) { // ha megtalálta a táblát
        $row = $result->fetch_assoc();
        if ($password === $row['password']) { // Ha a jelszó megegyezik irányítson át az
index.phpra
            $_SESSION['login_user'] = $username;
            header("location: index.php");
            exit();
        } else {
            $error_message = "Your Username or Password is invalid";
        }
    } else {
        $error_message = "Your Username or Password is invalid";
    }
}
```

MAGA A WEBOLDAL:

```
<header>
   <div class="content">
      <h1>Rólunk</h1>
      <nav>
         <ul>
            <li><a href="index.html">Főoldal</a></li>
            <li><a href="download.html">Letöltés</a></li>
            <li><a href="caraters.html">Karakterek</a></li>
            <li><a href="about.html">Rólunk</a></li>
            <li><a href="profil.php">Profil</a></li>
         </ul>
      </nav>
   </div>
</header>
```

Ez a HTML kód részlet egy egyszerű, hatékonyan strukturált weboldal fejlécét mutatja be. A "header" elem egy olyan konténert jelöl, amely az oldal tetején helyezkedik el, és a tartalmat, vagyis a fejlécet fogja tartalmazni. A "content" osztály segítségével a fejléc tartalma kerül elhelyezésre. Az "h1" címsorban található "Rólunk" szöveg az oldal rövid leírását vagy címsorát jelöli. A navigációs menü egy "nav" elemen belül található, ami megfelelő strukturáltságot biztosít a menünek. Az "ul" listában található "li" elemek a menüpontokat tartalmazzák, amelyek "a" hivatkozásokat tartalmaznak a különböző oldalakhoz. A kód azért van így strukturálva, hogy könnyen érthető legyen és könnyen kezelhető legyen a weboldal tartalma és navigációja szempontjából. A megfelelő elemek és osztályok használata segíti a fejlesztőket abban, hogy könnyen átlátható és karbantartható kódot hozzanak létre.

```
<main>
   <section id="story" class="content">
      <h2>Üdvözlünk a Flare Studio hivatalos weboldalán!</h2>
      <p><b>Merülj el a fantasztikus világunkban, ahol legyőzhetetlen hősök és
kalandok várnak rád!</b></p>
   </section>
   <section id="story" class="content">
      <div class="text">
         <h2>Játéktörténet</h2>
         <p><b>Sir Rodrik</b>szövegrész.</p>
         <p>szöveg</p>
         <p>szöveg.</p>
      </div>
      <div class="image">
         <img src="OIG4 (1).jpg">
```

```
      </div>
   </section>
   <section id="story" class="content">
      <div class="text">
         <h2>Viking történelem</h2>
         <p>szöveg</p>
      </div>
      <div class="image">
         <img src="hajo.jpg">
      </div>
   </section>
   <section id="story" class="content">
      <div class="text">
         <h2>A vikingek kirajzásának okai és formái:</h2>
         <p>szöveg</p>
         <p>szöveg.</p>
         <p>szöveg.</p>
      </div>
      <div class="image">
         <img src="terkep.png">
      </div>
   </section>
   <section id="story" class="content">
      <div class="text">
         <h2>Dán királyok Anglia trónján:</h2>
         <p>szöveg</p>
      </div>
   </section>
</main>
```

Ez a kód egy strukturált HTML dokumentumot ír le, amely különböző szekciókba rendezett tartalmat tartalmaz, és képeket is megjelenít. Ami a weboldal fő tartalmi részét, a "main" elemet reprezentálja, amely több "section" elemet tartalmaz, mindegyik egy-egy témával kapcsolatos részletet mutat be. Minden "section" elem egy egyedi azonosítóval rendelkezik ("id="story") és egy "content" osztállyal, amely segíti a CSS stílusok alkalmazását és a tartalom formázását. Minden "section" rész egy témával kapcsolatos, és tartalmaz egy címsort ("h2") és a hozzá kapcsolódó szöveges tartalmat ("p" elemek). Néhány rész "text" osztállyal rendelkezik a szöveges tartalomnak, és másik "image" osztállyal a hozzá kapcsolódó képeknek. A tartalom struktúrája segíti az oldal felhasználóbarát és könnyen navigálható kialakítását, valamint a CSS stílusok könnyebb alkalmazását a különböző tartalmi részekre. Ezáltal a felhasználók könnyen megtalálhatják az érdeklődési területeiknek megfelelő információkat. Néhány "section" elem tartalmaz egy "text" osztályú szöveges tartalmat és egy "image" osztályú képet együtt. Ez lehetővé teszi a szöveges és vizuális információk hatékony kombinálását, ami

segíthet az olvasóknak a jobb megértésben és élményben. A `<footer>` elem azért van ott, hogy a weboldal alján megjelenítse az oldal tulajdonosának vagy készítőjének információit. Ebben az esetben a "welcome" osztály attribútummal van ellátva, ami lehetőséget ad arra, hogy a stíluslapban vagy a JavaScript kódban különféle formázási vagy viselkedési szabályokat alkalmazzunk rá. A `<footer>` elem tartalmazza a szerzői jogokkal kapcsolatos információt, amely azt mondja: "© 2024 Flare Studio. Minden jog fenntartva." Ez azért fontos, mert megvédi az oldal tartalmát, és tudatja a látogatókkal, hogy a tartalom tulajdonosa az adott stúdió, és bármilyen másolás vagy felhasználás csak az ő engedélyükkel lehetséges. Ez a szöveg gyakori lábjegyzet egy weboldalon, amely jogi védelmet nyújt az oldal készítőjének.

```
<!DOCTYPE html>
<html lang="hu">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Odin's Fury - Letöltés</title>
   <link rel="stylesheet" href="download.css">
</head>
<body>
   <header>
      <div class="content">
         <h1>Letöltés és irányítás</h1>
         <nav>
            <ul>
               <li><a href="index.html">Főoldal</a></li>
               <li><a href="download.html">Letöltés</a></li>
               <li><a href="caracters.html">Karakterek</a></li>
               <li><a href="about.html">Rólunk</a></li>
               <li><a href="profil.php">Profil</a></li>
            </ul>
         </nav>
      </div>
   </header>
   <main>
      <section id="controls" class="content">
         <h2>Irányítás</h2>
         <div>
            <h3>Billentyűzet mozgás:</h3>
            <p>W - előre</p>
            <p>A - balra</p>
            <p>S - hátra</p>
            <p>D - jobbra</p>
         </div>
         <div>
            <h3>Egér mozgás:</h3>
```

```
            <p>Húzd az egeret - jobbra</p>
            <p>balra</p>
            <p>hátra</p>
            <p>jobbra</p>
          </div>
          <div>
            <h3>Támadás:</h3>
            <p>bal kattintás</p>
          </div>
          <div>
            <h3>Interakció:</h3>
            <p>E</p>
          </div>
          <div>
            <h3>Tárgymenü:</h3>
            <p>P</p>
          </div>
        </section>
        <section id="locations" class="content">
          <h2>Helyszínek</h2>
          <ul>
            <li>Kietlen sivár vidék</li>
            <li>Falu</li>
            <li>Colosseum</li>
            <li>Királyi vár</li>
          </ul>
          <div class="image">
            <img src="helyszin.jpg">
          </div>
        </section>
        <section id="shops" class="content">
          <h2>Boltok</h2>
          <ul>
            <li>Fegyver bolt - vegyél a boltban felszerelés, hogy még erősebb legyél és a
védelmed is!</li>
            <li>Varázsló bolt - szerezd be a Földön túli képességet!</li>
          </ul>
          <div class="image">
            <img src="shops.jpg">
          </div>
        </section>
        <section id="games" class="content">
          <h2>Játékok</h2>
          <ul>
            <li>Aréna mód: a Colosseumban bemutathatod, hogy meddig tudsz talpon
maradni!</li>
            <li>Quiz: mutasd meg képességeidet, hogy te lehess a legjobb!</li>
            <li>Blackjack: Tedd próbára a korabeli szerencsejátékot (Csak saját
felelősségedre!)</li>
          </ul>
          <div class="image">
            <img src="jatekok.jpg">
```

```
            </div>
        </section>
        <section id="download" class="content">
            <h2>Letöltés</h2>
            <p>Kérlek, kattints a gombra az Odin's Fury letöltéséhez:</p>
            <a href="https://github.com/your-
username/OdinsFury/archive/refs/heads/main.zip" class="download-
button">Letöltés</a>
        </section>
    </main>
    <footer class="welcome">
        <p>&copy; 2024 Flare Studio. Minden jog fenntartva.</p>
    </footer>
    <script src="download.js"></script>
</body>
</html>
```

Ez a HTML kód egy fiktív játék, az "Odin's Fury" letöltési oldalát leíró struktúrát mutatja be. Az oldal célja, hogy bemutassa a játék letöltésének lehetőségét, valamint az irányítását, helyszíneit, boltjait és játékait. Fejléc (head): Az oldal fejlécében található meta tagok beállítják az oldal karakterkódolását és a nézettérfogatot. Emellett a cím meghatározza az oldal címét, ami az "Odin's Fury - Letöltés". A külső stíluslap hivatkozása a megjelenítésért felelős. Törzs (body): Az oldal törzsében található elemek között szerepel a fejléc, amely a címet és a navigációs menüt tartalmazza. A fő tartalom (main) részben különböző szekciók vannak, amelyek az irányítást, helyszíneket, boltokat, játékokat és a letöltés lehetőségét mutatják be. A lábléc (footer) alatt található meg a szerzői jogokat és a stúdió nevét jelző szöveg. A képek és linkek segítségével az oldal színes és interaktív módon jeleníti meg a játékhoz kapcsolódó információkat, valamint lehetőséget biztosít a játék letöltésére. A felhasználók könnyen navigálhatnak az oldalon keresztül a menüpontok segítségével, és részletes információkat kaphatnak a játékról és annak funkcióiról. Az oldal érdekesebb részei közé tartozik az "Irányítás" szekció, amely részletesen bemutatja, hogyan lehet a játékot irányítani. A szöveges leírás mellett a szekció különféle billentyűkombinációkkal és az egérmozgással kapcsolatos információkat is tartalmaz, ami segít a játékosoknak megtanulni a játékvezérlést. A "Helyszínek" és "Boltok" szekciók szintén érdekesek lehetnek, mivel bemutatják a játék világát és lehetőségeit. A "Helyszínek" rész felsorolja a különböző játékkörnyezeteket, amelyekben a játékosok felfedezhetnek és küzdenek. A "Boltok" szekcióban pedig megismerhetjük, hogy milyen felszerelések és varázslatok érhetők el a játékban, és hogyan segíthetik elő a játékosok fejlődését. Az "Aréna mód" és a "Quiz" játékok bemutatása a "Játékok" szekcióban izgalmas lehet, mivel ezek különleges játékmódokat és kihívásokat jelentenek a játékosoknak. Az "Aréna mód" arra ösztönzi a játékosokat, hogy talpon maradjanak a Colosseumban, míg a "Quiz" lehetőséget ad arra, hogy a játékosok megmutassák tudásukat és képességeiket. Végül, az oldal alján található "Letöltés" szekció a legfontosabb, mivel lehetőséget biztosít a játék letöltésére. A letöltési gomb egyenesen a letöltési forráshoz vezet, ami gyors és könnyű hozzáférést biztosít a játékhoz. Ez az érdekesebb részek közé tartozik, mert a látogatók itt kapják meg azonnali lehetőséget a játékhoz való hozzáférésre.

```
<!DOCTYPE html>
<html lang="hu">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Odin's Fury - Rólunk</title>
   <link rel="stylesheet" href="about.css">
</head>
<body>
   <header>
      <div class="content">
         <h1>Rólunk</h1>
         <nav>
            <ul>
               <li><a href="index.html">Főoldal</a></li>
               <li><a href="download.html">Letöltés</a></li>
               <li><a href="caracters.html">Karakterek</a></li>
               <li><a href="about.html">Rólunk</a></li>
               <li><a href="profil.php">Profil</a></li>
            </ul>
         </nav>
      </div>
   </header>
   <main id="kozep">
      <section id="story" class="content">
         <div class="text">
            <h2>Miért csináljuk a játékot?</h2>
            <p>Szöveg.</p>
         </div>
      </section>
      <section id="story" class="content">
         <div class="text">
            <h2>Mikor kezdtük el?</h2>
            <p>Szöveg.</p>
         </div>
      </section>
      <section id="story" class="content">
         <div class="text">
            <h2>Nehézségek és előnyök</h2>
            <p>szöveg</p>
         </div>
      </section>
      <section id="story" class="content">
         <div class="text">
            <h2>"Pénzügyi befektetés"</h2>
            <p>A projektet kezdetben és továbbra is az ingyenes alapanyagok
segítségével hoztuk létre.</p>
         </div>
      </section>
      <section id="story" class="content">
         <div class="text">
```

```
            <h2>Feladatok és elkészítés</h2>
            <p>szöveg.</p>
          </div>
        </section>
    </main>
    <footer class="welcome">
        <p>&copy; 2024 Flare Studio. Minden jog fenntartva.</p>
    </footer>
    <script src="index.js"></script>
</body>
</html>
```

Ez a HTML kód egy teljes weboldalt reprezentál, amelynek célja bemutatni a "Odin's Fury" nevű játékot és annak fejlesztőit. Az oldal strukturált módon van megírva, a HTML elemek jól definiáltak és könnyen értelmezhetők. Az oldal tartalmazza a fejlécet, amely a navigációs menüt és az oldal címét tartalmazza, valamint a fő tartalmi részt, amely részletesen bemutatja a játék projektet és annak hátterét. A "main" elem tartalmazza a fő tartalmat, amelynek részei különböző szekciók, mindegyik egy-egy témával kapcsolatos információt tartalmaz. Az oldal végén található a lábléc, amely a szerzői jogi információkat tartalmazza. Emellett a kód hivatkozik egy külső CSS fájlra a stílusok alkalmazásához, valamint egy JavaScript fájlra a dinamikus funkcionalitásért. A kód ilyen struktúrában van lekódolva, hogy könnyen karbantartható legyen, és az információk logikus módon legyenek elrendezve az oldalon. A strukturált kódolás és a megfelelő HTML elemek használata lehetővé teszi a könnyű stílusok alkalmazását és a dinamikus funkcionalitás integrálását, ami javítja a felhasználói élményt és az oldal hatékonyságát.

```
<?php
session_start();
include 'db.php'; // database meghívása
 // visszavisz a loginra ha nem vagy bejeléntkezve
if (!isset($_SESSION['login_user'])) {
    header("location: login1.php");
    exit();
}
 // Fetch a felhasználó adatait
$username = $_SESSION['login_user']; // meghívás a sessionre
$query = "SELECT coins FROM unitybackendtutorial WHERE username = ?";//lekérés
adatbázisból
$stmt = $conn->prepare($query);
$stmt->bind_param("s", $username);
$stmt->execute(); // Executes the query
$result = $stmt->get_result(); // az eredmény
 if ($result->num_rows > 0) {
    $row = $result->fetch_assoc();
    $coins = $row['coins'];
} else {
    $coins = "No data available"; //ha nincs adat
}
```

```
$stmt->close(); // bezárás

// a top 10 usernak a a leaderboardja
$query2 = "SELECT username, level FROM `unitybackendtutorial` ORDER BY level
DESC LIMIT 10";
$stmt2 = $conn->prepare($query2);
$stmt2->execute();
$result2 = $stmt2->get_result();

$topUsers = []; // tömb a felhasználóknak
while ($row = $result2->fetch_assoc()) {
    $topUsers[] = $row; // hozzáadás
}
$stmt2->close(); //

// a szint lekérése
$query1 = "SELECT level FROM unitybackendtutorial WHERE username = ?";

$stmt = $conn->prepare($query1);
$stmt->bind_param("s", $username);
$stmt->execute();
$result = $stmt->get_result();


if ($result->num_rows > 0) {
    $row = $result->fetch_assoc();
    $level = $row['level'];
} else {
    $level = "No data available"; // ha nincs data
}
$stmt->close();
?>
```

Ebben a php-ban lekérjük a eltároljuk egy sessionbe a felhasználó adatait majd lekérdezzük mind a coins meg a levelt a felhasználótól és kiiratjuk egy listába illetve van itt egy leaderboard ahol a top 10 usert találjuk akik már ennyiszer kivitték a játékot.

Felhasználói dokumentáció:
A bejelentkezés után amit ezen a linken keresztül lehet megtenni:
https://mukodj123.000webhostapp.com/login1.php
A download/ letöltés fülre kattintva ott található a letöltés link, ha arra rákattintunk bedob a github nevezetű oldalra ahol zip fájlba lelehet tölteni az egészet, majd azt kibontva láthatunke egy olyan zipet hogy build 15,azt is kibontva láthatunk egy build 15 mappát és ott található egy NewUnityProject nevű fájl arra rákattintva betöltünka játékba. Ott az intro után betudunk jelentkezni / Regisztrálni azután ha sikeres választhatunk hogy Story Mode vagy Battle , ha a story Modet választjuka az intro után bedob a tengeri csatára A illetve D vel lehet balra jobbra mozogni q val meg e vel lőni és miután megöltük a 4 wavet akkor ha lenyomjuk a spacet áttob a csatamezőre. Ha végigért a hajó bemehetünk a colosseumba ahol a 4 wave kiirtása után tovább mehetünk vagy harcolhatunk tovább több goldért. Kiértünk balra akkor a kunyhókba különböző segítségeket találhatunk: bolt, blackjack, logical, magic. Ha bevásároltunk sétálhatunk tovább a kastélyba ahol megkell ölnünk a tank karaktereket és az íjászokat hogy végül legyőzhessük a boss-t.

Összefoglaló: Összeségében gazdagodtunk ezzel a projectel. Megtanultunk együtt dolgozni egy csapatba, habár voltak nehézségek, de sikerült. Habár úgy érzem lehetett volna jobb is a játék de összeségében elégedett vagyok vele mert szerintem nem lett annyira rossz. Öröm volt látni, hogy a programozási képességeinknél, hogy mennyit milyen gyorsan fejlődött, már bánom, hogy nem volt egy kicsivel több időnk. Végszónak annyi, hogy a nehézségek ellenére is jó élmény volt.