VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

GRAFINIŲ SISTEMŲ KATEDRA KATEDRA

Kompiuterinių žaidimų kūrimo pagrindai

Kursinis darbas

Atliko: MKDF-21/1 gr. studentai Nedas Čepaitis,

Ernestas Beniušis,

Lukas Vasiliauskas

Tikrino: Doc. Dr. Girūta Kazakevičiūtė-Januškevičienė

VILNIUS 2024

Turinys

Kompiuterinių žaidimų kūrimo pagrindai 1
Turinys 2
1. Žaidimo pavadinimas, žanras ir aktualumas 3
2. Auditorija 3
3. Sukurti tokio tipo žaidimai 4
4. Žaidimo tikslas ir scenarijus 5
5. Žaidimo kūrimo etapų pagrindiniai aspektai, mechanika ir intelektika 6
5.1 Žaidimo objektai, jų komponentai bei skriptai 6
5.2 Naudotojo sąsaja 13
6. Žaidimo lygiai ir jų progresyvumas 14
7. Žaidimo platinimas 17
8. Išvados 17
9. Naudoti literatūros šaltiniai 18
10. Unity versija 18

1. Žaidimo pavadinimas, žanras ir aktualumas

Žaidimo pavadinimas: Duck Frenzy.

Žanras: Psichologinis siaubo žaidimas.

Aktualumas: Siaubo žaidimų žanras yra vienas iš patraukliausių vaizdo žaidimų žanrų. Ir suvis kiekvienais metais, šio žanro žaidimų, daugėja įvairiose internetinėse ir fizinėse vaizdo žaidimų parduotuvėse.

Siaubo žaidimuose yra gausu įvairios kraupios medžiagos, ar tai būtų kraujas, ar mirę herojai/kiti personažai, ar nutrauktos galūnės. Psichologinis siaubas nepasižymi labai akivaizdžia kraupia medžiaga.

Psichologinis siaubas yra vystomas nuo pirmojo žaidimo momento ir jam progresuojant, psichologinis siaubas arba yra vystomas kartu su siužetu, arba tiesa yra tiek užglaistoma melu, jog pats siaubas tampa pastebimas tik žaidėjui artėjant prie žaidimo pabaigos.

2. Auditorija

Auditorijos nustatymas dažnai vyksta dar net neiškėlus pagrindinės žaidimo idėjos. Tačiau, taip pat dažnai, žaidimo žanras ir jo auditorija susiformuoja bekuriant žaidimą.

Virš 18 metų amžiaus apribojimas buvo pasirinktas pagal PEGI reitingavimo sistemą, kur PEGI 18 - Skirta tik suaugusiems asmenims. Šis apribojimas buvo paskirtas, kadangi žaidime vyksta bemotyvis žudymas, beginklių asmenų/gyvūnų žudymas.

3. Sukurti tokio tipo žaidimai

Egzistuoja tikrai nemažai psichologinio siaubo žanro žaidimų, tačiau yra keli labai populiarūs žaidimai, tokie kaip Doki Doki Literature Club, Inscryption.

Doki Doki Literature Club – tai žaidimas, kuriame pagrindinis veikėjas turi taikstytis su mergina kuri turi galių laužyti “4-tą sieną” (gebėjimas matyti už žaidimo erdvės ribų, kai “4-ta siena” yra žaidėjo ekranas)

Inscryption – tai vienas iš retai populiariais tampančių “Game Jam’ų” žaidimų. Tais metais, kai žaidimas dalyvavo šiame renginyje, visiems konkurentams buvo iškelta tema “Aukos turi būti atliktos” (“Sacrifices must be made”). Šis žaidimas yra eilių (turn based) žaidimas, kuriame reikia atsikratyti prastesnių, tam tikrų gyvūnų kortelių, dėl stipresnės gyvūno kortelės. Kiekvieną kartą padarius tokią auką, žaidėas turi susitaikyti su, galbūt, jam patinkančios ir ilgai išlaikytos gyvūno kortelės praradimu.

Aspektai, kurie buvo tobulinti ir idėti į žaidimą:

Vienas pagrindinių aspektų, kurį buvo nuspręsta tobulinti žaidime "Ducky Frenzy", yra dinaminė emocijų simuliacija, kuri įtakoja žaidimo eigą ir žaidėjo patirtį. Šis aspektas apima kelis elementus:

- Dialogai ir Interakcija su Duck'u:

Duck’as nėra tik statinis objektas, jis bus kaip veikėjas, su kuriuo bus šnekama žaidimo metu. Žaidimo metu šuo Duck'as keis savo bendravimą ir taps vis grubesnis ir piktesnis, o tai kuria jausmą, kad Duck’as yra gyvas personažas su tikrais jausmais ir intencijomis.

- Ančių Būsenos ir Variacijos:

  Ančių išvaizda ir elgesys kinta priklausomai nuo to, kaip dažnai jos buvo pašautos. Kruvinos ir sužeistos antys sukelia stipresnį emocinį atsaką, sukuriant intensyvesnę ir įtemptesnę atmosferą.
  Skirtingos ančių spalvos ir būsenos ne tik pagyvina vizualinį žaidimo aspektą, bet ir suteikia papildomų strateginių galimybių žaidėjui.

- Žaidėjo Psichologinė Būsena:

  Žaidimas simuliuoja žaidėjo psichologinę būseną, priversdamas žaidėją abejoti savo veiksmais ir pasirinkimais. Duck’as gali manipuliuoti žaidėjo emocijomis, priversdamas jį jaustis kaltu už ančių šaudymą arba didžiuotis dėl pasiekimų.
  Ši emocinė dinamika skatina žaidėją gilintis į žaidimą ir asmeniškai įsitraukti į istoriją.

- Gilesnis Įsitraukimas:

  Dinaminė emocijų simuliacija leidžia žaidėjams labiau įsijausti į žaidimo pasaulį ir jaustis asmeniškai įtrauktiems į istoriją. Emocinis ryšys su Duck’u ir kintančios ančių būsenos sukuria unikalią ir intensyvią patirtį.

  Tobulindami šiuos aspektus mes sukursime ne tik pramogą, bet ir emocinį iššūkį, kuris skatintų žaidėjus gilintis į žaidimo pasaulį ir patirti įvairias emocines būsenas.

4. Žaidimo tikslas ir scenarijus

Tikslas: Nušauti kiek įmanoma daugiau ančių ir surinkti kiek įmanoma geresnį rezultatą, arba pagerinti savo buvusį aukščiausią rezultatą, bei išsiaiškinti, kas iš tikro yra tas Duck‘as.

Scenarijus: Šiame žaidime, Duck’as yra apleistas šuo, kuris visa savo gyvenimą negavo nei lašelio meilės. Būdamas jaunas ir naivus, herojus susidraugauja su benamiu šunimi. Jis leidžia kartu laiką su juo, greitu metu suranda karnivalinį žaidimą “Ducky Frenzy”.

Duck’as bandys tave įtikinti, jog karnavalinis ančių šaudymas tėra tik žaidimas. Norėdamas išpildyti savo keistas genocidines idėjas, jis tave vers pabandyti tai daryti ir už karnivalinio žaidimo ribų.

Greitu metu, būdamas naiviu patikliu jaunuoliu, nebegalėsi nuspręsti į ką šausi ir į ką ne, kadangi Duck’as nebebus tik draugas, jis bus tavo vergvaldys.

5. Žaidimo kūrimo etapų pagrindiniai aspektai, mechanika ir intelektika

5.1 Žaidimo objektai, jų komponentai bei skriptai

a) Žaidimo objektus sudaro: Scena, antys, ginklas, šuo Duck’as.

Kadangi visas žaidimas bus pikseliuotas, visi objektai bus kuriami mūsų pačių, arba bus perpiešti iš internete įkeltų įkvepiančių nuotraukų.

Antims, ginklui ir šuniui bus sukurtos kelios būsenos/variacijos, kurios žaidimą padarys atrodantį daug aktyvesnį.

Įvairovės dėlei, antys turės kelias būsenos, kurios priklausys nuo tam tikros spalvos anties pašovimų skaičiaus (kuo daugiau kartų antis yra pašauta, tuo kruvinesnė ji atrodo)

Žaidime kamera ir šviesos bus statinės, nekintančios. Šviesa mes šešėlį ant galinės sienos.

b) Žaidime bus keli statiniai objektai, kaip šuo Duck‘as, kuris turės savo dialogą ir šnekės su mumis kaip žaidėju. Kitas statinis objektas – pagrindinė scena.Judantys objektai – antys ir ginklas.

Visi objektai turės Sprite’ams reikalingus komponentus, kad jie būtų teisingai rodomi ekrane.

Kadangi žaidimas yra iš esmės gana paprastas, jam užtenka parašyti pakankamai skriptų, jog viskas veiktų atitinkamai. Unity variklis neturi daugiau mums naudingų komponentų.

Keletas kodo fragmentų

```
if (duckSpawner.spawnerDuckDirection == DuckSpawner.DuckDirection.Left)
    transform.position = new Vector3(transform.position.x - x, transform.position.y, transform.position.z);
else
    transform.position = new Vector3(transform.position.x + x, transform.position.y, transform.position.z);

if (
        (transform.position.x < -8f && duckSpawner.spawnerDuckDirection == DuckSpawner.DuckDirection.Left) ||
        (transform.position.x > 8f && duckSpawner.spawnerDuckDirection == DuckSpawner.DuckDirection.Right)
    )
{
    if (!isShot && !isChild)
    {
        if(FindAnyObjectByType<GunShootsDuck>().score>0 && !FindAnyObjectByType<DialogManager>().dialogStarted)
        FindAnyObjectByType<GunShootsDuck>().score--;
    }
    Destroy(this.gameObject); // when reached edge - destroy it

}
```

Anties judėjimas. Greitis x yra nustatomas priklausomai ar antis yra vaikas, ar ne. Pasiekus kraštą antis yra sunaikinama (kad neapkrauti sistemos), o jei ji nebuvo pašauta, žaidimo taškai yra atemami

```
if ((lastDuckSpawned == null || Vector3.Distance(lastDuckSpawned.transform.position, spawnerTransform.position) > 4f) && Equals(lastDuckSpawned))
{
    int random0Antis = UnityEngine.Random.Range(0, antys.Length);
    GameObject selectedAntis = antys[random0Antis];


    GameObject newAntis = Instantiate<GameObject>(selectedAntis, spawnerTransform.position, spawnerTransform.rotation);
    newAntis.GetComponent<DuckMovement>().duckSpawner = this;
    newAntis.GetComponent<DuckMovement>().duckDirection = spawnerDuckDirection;

    lastDuckSpawned = newAntis;

    SpawnDuckies(newAntis, selectedAntis);
}
```

Antis yra instancijuojama, jeigu tarp paskutinės instancijuotos anties yra 4-ių vienetų atstumas. Antis yra parenkama iš kiekvienoje scenoje pridėtų ančių Prefab'ų.

```
if (!hitObject.GetComponent<DuckMovement>().isShot && !hitObject.GetComponent<DuckMovement>().isChild)
{
    if (hitObject.GetComponent<DuckMovement>().isThrowable)
    {
        thrownDucksShot++;
        score += 15;
    }
    else
        score++;

    gameManager.probabilityOfThrownDuck += 0.03f;
    gameManager.randomFloat = gameManager.GetRandFloat();

    hitObject.GetComponent<DuckMovement>().isShot = true;

}
```

Naudojant Polygon 2D kolaiderius yra nustatomi kokie objektai scenoje yra antys, jei antis yra nušaunama tuomet pridedami taškai. Yra nustatoma "pašautos" anties būsena, kad būtų blokuojami kai kurie veiksmai, judesiai, algoritmai.

```
if (!dialogStarted && nextDialogIndex < scoreThresholds.Length && !dialogBoxInMotion)
{
    int score = gunShootsDuckScript.score; // Get the score from the GunShootsDuck script

    // Check if a new dialog should be initiated based on the score
    if (score >= scoreThresholds[nextDialogIndex] && dialogFinished)
    {
        StartDialog(nextDialogIndex);
        nextDialogIndex++; // Move to the next dialog index
    }
}
```

Pasiekus tam tikrą taškų skaičių yra iškviečiamas dialogas. Išlenda dialogo langas bei šuo.

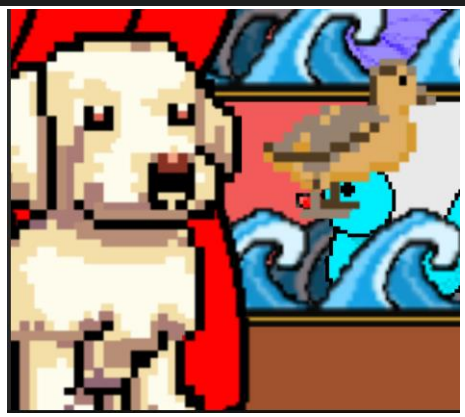
```
IEnumerator TypeText(string line)
{
    isTyping = true;
    dialogText.text = "";
    foreach (char letter in line.ToCharArray())
    {
        dialogText.text += letter;
        yield return new WaitForSeconds(0.05f); // Adjust the typing speed as needed
    }
    isTyping = false;
}
```

Naudodami coroutine yra rašomas dialogo tekstas po vieną raidę.

```
if (dogAnimator.GetCurrentAnimatorStateInfo(0).normalizedTime >= 0.5f && !duckThrown && dogAnimator.GetCurrentAnimatorStateInfo(0).IsName("DogMoves"))
{
    // Unparent the duck from the dog
    duck.transform.SetParent(duck.transform.parent.transform.parent, true);
    duck.transform.position = dogAnimator.transform.position;
    // Trigger the "Middle" animation of the duck
    duckAnimator.SetTrigger("Middle");
    // Set duckThrown to true to avoid repeating this process
    duckThrown = true;
}

if (duckAnimator.GetCurrentAnimatorStateInfo(0).normalizedTime >= 0.9f && duckThrown && duckAnimator.GetCurrentAnimatorStateInfo(0).IsName("ThrowableDuckAnimation"))
{
    // Unparent the duck from the dog
    duck.transform.SetParent(dogAnimator.transform, true);
    // Trigger the "Middle" animation of the duck
    duckThrown = false;
    duck.GetComponent<DuckMovement>().isShot = false;
}
```

nušovus tam tikrą skaičių ančių yra galimybė šuniui išmesti tikrą antį už kurią bus gaunama daugiau taškų (15 taškų).



Lygiui pasibaigus yra užsklendžiamos užuolaidos ir pereinamas į pabaigto lygio scena, kurioje rodomi taškai, kiek kartų buvo iššauta ir kiek tikrų ančių buvo nužudyta.

```
public IEnumerator FadeOutScreen()
{
    float duration = 2.0f; // Duration in seconds
    float currentTime = 0f;
    Color originalColor = fadeOverlay.color;
    Color targetColor = new Color(originalColor.r, originalColor.g, originalColor.b, 1);

    while (currentTime < duration)
    {
        currentTime += Time.deltaTime;
        float alpha = Mathf.Lerp(0, 1, currentTime / duration);
        fadeOverlay.color = new Color(originalColor.r, originalColor.g, originalColor.b, alpha);
        yield return null;
    }
}
```

Užtamsinama scena, po to kai yra užtraukiamos užuolaidos. Užuolaidos yra užtraukiamos, kai pasiekiamas tam tikras skaičius taškų

```
if (!gameManager.dialogManager.dialogStarted)
{
    List<float> probabilities = new List<float>();
    foreach (Animator animator in animators)
    {
        RectTransform animatorRect = animator.GetComponent<RectTransform>();
        float distance = Vector2.Distance(animatorRect.position, crosshair.position);
        //Debug.Log(animator.name + " : " + distance);
        float probability = distance; // Inverse scaling
        probabilities.Add(probability);
    }

    // Select animator based on probabilities
    int selectedIndex = SelectAnimator(probabilities);

    // Trigger Move in the selected animator
    for (int i = 0; i < animators.Count; i++)
    {
        if (i == selectedIndex)
            animators[i].SetTrigger("Move");

        //Debug.Log("selected " + selectedIndex);
    }
}
```

Yra pamatuojami atstumai tarp taikinuko ir visų šūnų esančių už medžių boso lygyje. Pagal artimiausią atstumą yra parenkamas šuo kuris keliaus atsiimti nušautos anties.

```
float sum = 0;
int a = 0;
foreach (float probability in probabilities)
{
    sum += probability;
    //Debug.Log(animators[a].name + " : probability : " + probability);
    a++;
}
for (int i = 0; i < probabilities.Count; i++)
{
    probabilities[i] /= sum;
    //Debug.Log(animators[i].name + " : probability : " + probabilities[i]);
}

// Select animator based on probabilities
float randomValue = Random.value;
float cumulativeProbability = 0f;
for (int i = 0; i < probabilities.Count; i++)
{
    cumulativeProbability += (probabilities[i]);
    if (randomValue <= cumulativeProbability)
    {
        return i;
    }
}

// This should never happen, but just in case
return probabilities.Count - 1;
```

pagal atstumą yra nustatoma tikimybė tam tikram šuniui būti parinktam pajudėjimui iš už medžio (kuo didesnis atstumas, tuo didesnis šansas išlysti), kad žaidėjas galėtų jį nušauti.

```
InvokeRepeating("MoveDuck", 0, 0);

if (
      (rectTransform.anchoredPosition.x < (float)-80 && duckSpawner.spawnerDuckDirection == DuckSpawnerBossLevel.DuckDirection.Left) ||
      (rectTransform.anchoredPosition.x > (float)2000 && duckSpawner.spawnerDuckDirection == DuckSpawnerBossLevel.DuckDirection.Right) &&
      (!isTouchingGround)
   )
{
   Destroy(this.gameObject); // when reached edge - destroy it
gameManager.spawnTime = Time.time;

}

if (isShot && !isTouchingGround && !hasTouchedGround)
{

   GetComponent<Animator>().enabled = false;
   GetComponent<UnityEngine.UI.Image>().sprite = gameManager.fallingSprite;

   rectTransform.anchoredPosition = new Vector2(rectTransform.anchoredPosition.x, rectTransform.anchoredPosition.y - verticalSpeed * Time.deltaTime);
   verticalSpeed += verticalSpeed * 3 * Time.deltaTime;

   rectTransform.localRotation = Quaternion.Euler(new Vector3(180, 0, -Mathf.Atan2(rectTransform.anchoredPosition.y - lastPos.y, rectTransform.anchoredPosition.x - lastPos.x) * 180 / Mathf.PI));
}

//MISSING REFERNCE TO COMPONENT FIX IT NOW TODO

if (rectTransform.anchoredPosition.y < 220f)
{
   isTouchingGround = true;
}
```

Pagal atsitiktinį skaičių yra parenkamas spawneris kuris instancijuos skrendančią antį, pagal skridimo trajektoriją yra valdomas jos judėjimas. Antis gali būti pašauta, kai ji yra pašaunama, ji pradeda kristi, ir jos kritimo kampas yra skaičiuojamas pagal atan^2 funkciją.

```
    if (health <= 0)
    {
        health = 0;
    }

    healthBarTransform.localScale = new Vector3(health, 1, 1);
    if (health < 1.0f && health != 0)
        health += 0.01f * Time.deltaTime;

    currentColor = health * colorMin;

    foreach (var animator in dogAnimators)
    {
        animator.GetComponent<RawImage>().color = new Color(colorMin, currentColor, currentColor, 1);


    }

}
0 references
public void SetMaxHealth(float maxHp)
{
    healthBarTransform.localScale = new Vector3(maxHp, 1, 1);
}

1 reference
public void SetHealth(float damage)
{
    health -= damage;
}
```

Kiekvieną kartą pašovus šunį, jo gyvybės mažėja, priklausomai nuo gyvybių skaičiaus ir kiek kartų buvo pašautas šuo, yra paleidžiamas dialogas. Gyvybės lėtai atsistatinėja, kol šuo nėra pašautas 7 kartus arba gyvybės yra žemiau 40%

5.2 Naudotojo sąsaja

Preliminariai, žaidimas prasidės su pradiniu langu, kuriame žaidėjas galės pradėti/pratęsti žaidimą, pasikofigūruoti žaidimą jo norams (Nustatymų mygtukas), išjungti žaidimą.

Pačiame žaidime kaip vartotojo sąsajos dalis bus taškų sistema, kuria naudosis žaidimo logika žaidimo vystymui (Pasiekus tam tikrą skaičių taškų, žaidimas progresuos, taps sunkesnis, bus pristatomos naujos mechanikos kaip ančių skridimas)

Žaidžiant, pelytės judėjimas bus lydimas taikinuko.

Duck’o ir žaidėjo dialogas vyks taip pat per naudotojo sąsają, paspaudžiant kairyjį pelytės mygtuką, bus galima pereiti prie kitos dialogo eilutės, arba bus galima praleisti eilučių spausdinimą.

6. Žaidimo lygiai ir jų progresyvumas

Šiame žaidime bus trys lygiai, kuriuose herojus šaudys į plaukiančias antis karnivale bei kartais praskriejančias antis. Lygis baigiasi, kai žaidėjas surenka tam tikrą skaičių taškų.

Pirmame lygyje, herojus šaudys antis, ir dėl to jokių pasekmių nesulauks. Netgi priešingai, jis gaus apdovanojimus ir paskatinimus iš Duck'o – šuns.

Antrame lygyje bus matoma, kad antys pasikeitė, ir yra sužalotos ir kraujuotos, bei matomai, mirusios. Duck’as taps vis labiau brutalus, jis nori vis daugiau, tačiau po šiek tiek laiko, jis pradės bijoti mūsų žaidimo herojaus. Jis skatins žaidėją nužudyti vis daugiau ir daugiau ančių, tačiau elgsis atsargiai, kad netaptų žmogaus auka.

Paskutinis - boso lygio vaizdas.

Galutiniame lygyje herojus bando atsikratyti savo sadistiško draugo – šuns. Miške, šuo slepiasi už medžių. Viskas atrodo tikra, o Duck‘as save pradeda matyti ne kaip vergvaldį o auką.

Žaidėjas gauna užduotį nušauti Duck‘ą – draugą patapusį priešu. Apačioje ekrano yra matomos Duck‘o gyvybės, o gyvybėms išsekus, žaidimas pasibaigia su žinute „Tu laimėjai?“, lyg klaustų žaidėjo, ar jis tikrai **laimėjo**.

7. Žaidimo platinimas

Planuojame išleisti šį nuostabų žaidimą pačioje populiariausioje platformoje “Steam”, nes mūsų manymu, šioje platformoje žaidimas turėtų didžiausią galimybę sulaukti pripažinimo. Žinoma, norint, kad žaidimas sulauktų sekmės, jo reklamavimas yra vienas iš svarbiausių etapų, ir jis vyks platformose tokiose kaip X (Twitter), Facebook, Instagram, YouTube ir TikTok, žinoma, nepamiršime ir specialių akcijų internetinėje žaidimų parduotuvėje, bei nuolaidų tam tikrais sezonais.

8. Išvados

Darbas tikrai nebuvo vienas iš lengvųjų, prireikė tikrai daug laiko sukurti mūsų norimą žaidimą, išpildant mūsų – sau sukurtus lūkesčius, o kai kurių ir neįšėjus išpildyti.
Žaidimo programavimas sudarė labai didelę dalį viso laiko. Buvo labai sunku nepasimesti tarp kodų. Kitas iššūkis - išlaikyti objektų pirmenybę kuris objektas turi būti arčiau kameros, kuris toliau, kad tarkim taikinukas nepasislėptų už mygtuko. Taip pat, buvo ganėtinai sudėtinga įdiegti dialogus, kadangi jiems reikia atskiro judesio, o pajudėjus jiems tekstas turi būti rašomas, o kiti algoritmai sustabdyti tuo metu. Objektų kūrimas naudojant piskelart.com nesukėlė problemų, buvo ganėtinai paprasta tiktai pareikalavo truputuką kruopštumo bei išradingumo. Gautų 2D spraitų rezultatas visus tenkina.
Iškilo šiokių tokių neaiškumų eksportuojant žaidimą į galutinį rinkinį, dauguma žaidimo spraitų tapo baltais, tačiau galiausiai viską išėjo sutvarkyti.
Deja dėl laiko stokos ir įgūdžių trūkumo žaidime liko kai kurie, lengvai pastebimi bug‘ai.
Apibendrinant, iš šio projekto visi komandos nariai išsinešė tikrai nemažai žinių, ir darbas nors ir nebuvo lengvas, tikrai buvo visai įdomus ir naudingas.

9. Naudoti literatūros šaltiniai

How to make a sprite:
https://www.idtech.com/blog/how-to-make-a-sprite

EVERYTHING YOU NEED TO KNOW BEFORE YOU MAKE A 2D GAME
https://gamemaker.io/en/blog/how-to-make-a-2d-game

Guide to Making Better Horror Experiences
https://devforum.roblox.com/t/guide-to-making-better-horror-experiences/2212571/1

Naudota muzika

https://pixabay.com/music/search/genre/main%20title/

10. Unity versija

Žaidimas sukurtas naudojant Unity 2022.3.18f1 versiją. Ši versija pasirinkta dėl savo stabilumo ir naujausių funkcijų palaikymo, kurios padeda sukurti aukštos kokybės žaidimus.