Liceul tehnologic „Colegiul Școala Națională de Gaz", Mediaș

Proiect pentru susținerea atestatului profesional

FIREBOY & WATERGIRL IN THE FOREST TEMPLE

Profesor coordonator:

Sîrbu Margareta

Elev:

Micu Alexia Claudia

Cuprins

I. Motivul și scopul proiectului 3
II. Prezentarea generală a temei 3
III. Resurse hardware și software necesare 3
IV. Aspecte teoretice necesare realizării aplicației 4
V. Descrierea aplicației 5
A. Interfață 5
1. Meniul de start 5
2. Meniul de nivele 5
3. Nivele 6
4. Meniul de pauză 6
5. Meniul de final 7
B. Mecanici 8
1. Diamante 8
2. Apă, lavă și noroi 9
3. Bloc 9
4. Trapă 10
5. Buton 11
6. Manetă 12
7. Ușă 13
8. Jucători 14
C. Cod 18
1. Timer 18
2. StartMenu 18
3. Nivel eșuat sau terminat cu success 19
VI. Bibliografie și webografie 20

I. Motivul și scopul proiectului

Proiectul reprezintă atât punerea în aplicare a cunoștințelor învățate la școală, cât și posibilitatea combinării acestora cu pasiuni extracurriculare.

Realizarea acestui joc demonstrează cum un cod poate fi transformat într-o aplicație practică.

II. Prezentarea generală a temei

Fireboy & Watergirl este un joc multiplayer dezvoltat de Oslo Albet și apărut 2009, apreciat de generația tânără pentru abilitatea sa de a dezvolta munca în echipă într-un mod interactiv.

Jocul în Unity „Fireboy & Watergirl" este o reinterpretare în spațiul 3D a jocului original.

III. Resurse hardware și software necesare

Programele în care a fost construită aplicația sunt Unity Hub 2.4.5, Unity Version 2021.1, Visual Studio 2019 și Blender 2.93.3. Se deschide prin Unity Hub și Unity.

- Processor AMD Ryzen 5 3600 6-Core Processor 3.59 GHz
- Installed RAM 32.0 GB
- System type 64-bit operating system, x64-based processor
- Edition Windows 10 Pro
- Version 20H2
- OS build 19042.1526
- Size: 650 MB
- Size on disk: 674 MB
- Contains: 17.498 Files, 977 Folders

IV. Aspecte teoretice necesare realizării aplicației

Scopul este de a rezolva puzzle-urile care necesită atât implicarea jucătorului „Fireboy” (controlat prin ADW și Q2) cât și a jucătorului „Watergirl” (controlat prin săgeți și RightControl, RightShift). Cei doi trebuie să folosească manete, butoane, trape și ventilatoare pentru a ajunge la ușa fiecăruia, colectând diamante pe parcurs și având grijă să nu cadă în piscine periculoase (apă pentru Fireboy, lavă pentru Watergirl și noroi pentru amândoi).

La final, se acordă o notă bazată pe numărul de diamante colectate (A-verde, B-portocaliu, C-mov).

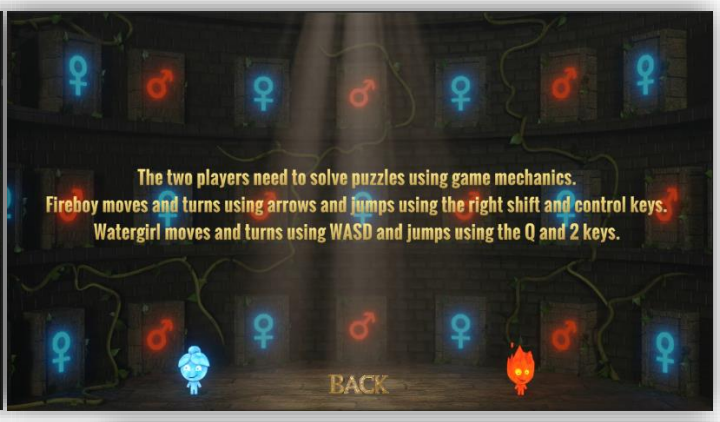
Cunoștințe necesare pentru realizarea proiectului:

- *(Visual Studio)* Programare în C# - un limbaj de programare cu uz general, cu mai multe paradigme. C# cuprinde scrierea statică, scrierea puternică, disciplinele de programare cu scop lexical, imperative, declarative, funcționale, generice, orientate pe obiecte și orientate pe componente.
- *(Unity)* Programarea pe obiect - Programarea orientată pe obiecte (OOP) este o paradigmă de programare bazată pe conceptul de „obiecte”, care poate conține date și cod: date sub formă de câmpuri (cunoscute adesea sub numele de atribute sau proprietăți), și cod, sub formă de proceduri (cunoscute adesea ca metode).
- *(Blender)*
    - Modelare 3d - În grafica computerizată 3D, modelarea 3D este procesul de dezvoltare a unei reprezentări bazate pe coordonate matematice a oricărei suprafețe a unui obiect (neanimat sau viu) în trei dimensiuni prin intermediul unui software specializat prin manipularea marginilor, vârfurilor și poligoanelor într-un spațiu 3D simulat. Modelele tridimensionale (3D) reprezintă un corp fizic folosind o colecție de puncte din spațiul 3D, conectate prin diverse entități geometrice precum triunghiuri, linii, suprafețe curbe etc. Fiind o colecție de date (puncte și alte informații), modelele 3D pot să fie create manual, algoritmic (modelare procedurală) sau prin scanare. Suprafețele lor pot fi definite în continuare prin maparea texturii, atribuirea unor materiale șamd.
    - Animare - „Rigging” (rig=animatură) este o tehnică folosită în animația scheletică pentru reprezentarea unui model de caracter 3D folosind o serie de oase digitale interconectate. Mai exact, „rigging” se referă la procesul de creare a structurii osoase a unui model 3D. Această structură osoasă este folosită pentru a manipula modelul 3D ca o marionetă pentru animație.

V. Descrierea aplicației

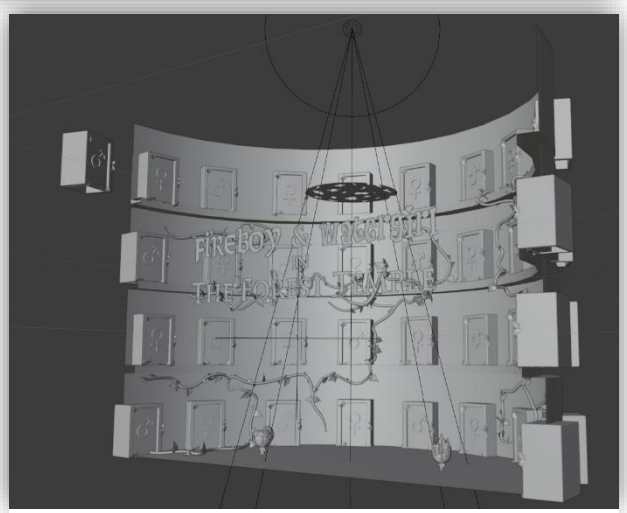
A. Interfață

1. Meniul de start



Aplicația începe cu interfața care conține elemente de UI organizate pe o componentă tip canvas:

- Trei butoane:
  - Play – conduce la meniul de nivele;
  - Instructions – afișează o explicație scurtă pentru a ajuta jucătorii;
  - Back – oprește rulajul.
- Videoclipul de pe fundal, care este făcut în aplicația 3d „Blender”;
- O coloană sonoră și efecte sonore asociate atingerii și alegerii butoanelor (hover + click).



Spațiul 3d din spatele videoclipului

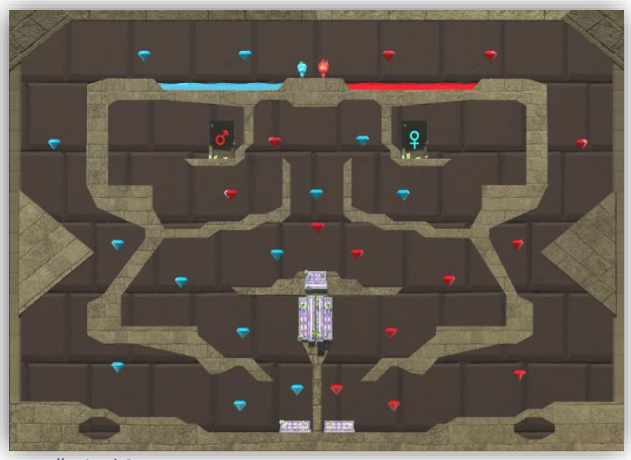
2. Meniul de nivele

Meniul de nivele conține alte trei butoane:

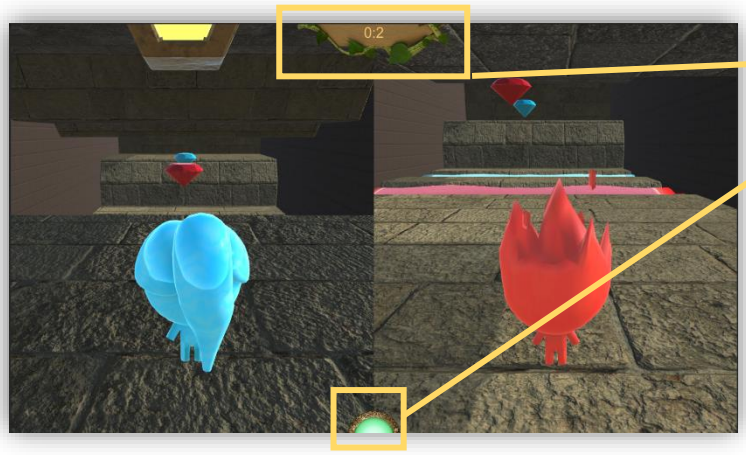
- Back – comută înapoi la Meniul de start;
- Două butoane (cu aspect de diamant) care conduc la nivele. Doar primul duce la un nivel terminat;
- O coloană sonoră și efecte sonore asociate atingerii și alegerii butoanelor și nivelelor.

3. Nivele

Hartă nivel 1



Hartă nivel 2



Timer

Buton care apelează meniul de pauză.

Ecranul este împărțit în două secțiuni pentru a face posibilă urmărirea amândurora jucătorilor în același timp.

Pe fundal este o coloană sonoră.

4. Meniul de pauză

Meniul de pauză (creat în Blender 3d) este asemănător celui care apare când un nivel este eșuat și are opțiunile:

- Restart – reîncepe nivelul;
- Map – duce la harta nivelului;
- Resume – continuă nivelul;
- Main Menu – duce înapoi la meniul de nivele.

Când meniul de pauză este activ, restul ecranului se întunecă, iar timer-ul se oprește.

5. Meniul de final

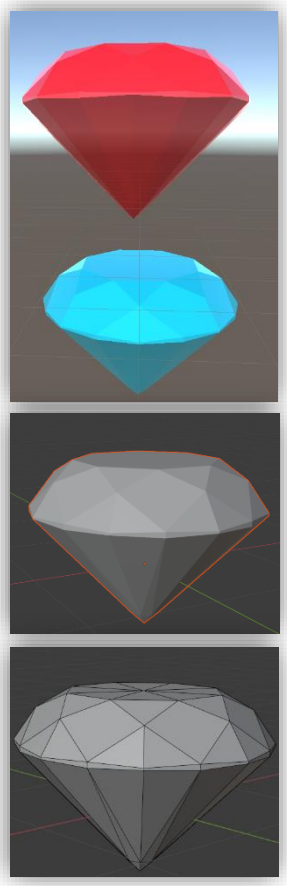
- Are efect sonor de „succes”;
- Afișează timpul;
- Afișează numărul de diamante de fiecare tip care au fost colectate pe parcursul jocului;
- Decide o notă pe baza diamantelor colectate prin funcția *DecideGrade* din script-ul *GemCollector;*

```
23      // call on finish level
        1 reference
24      public static void DecideGrade ()
25      {
26          if (LavaGemsCollected == LavaGemsTotal && WaterGemsTotal == WaterGemsCollected)
27              Grade = "A";
28          else
29              if (LavaGemsCollected == LavaGemsTotal || WaterGemsTotal == WaterGemsCollected && !Grade.Equals("A"))
30                  Grade = "B";
31              else
32                  if(Grade.Equals("O"))
33                      Grade = "C";
34      }
```

B. Mecanici

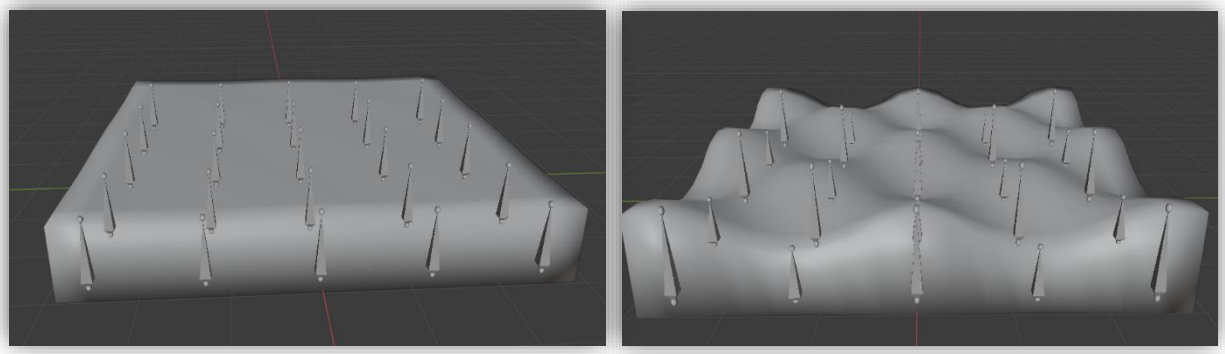
1. Diamante

Diamantele pot fi colectate de jucători (cel roșu de către Fireboy și cel albastru de către Watergirl). Acestea au efect sonor și se rotesc.



```
// gem collecting - jucatorii si diamantele au etichete asociate
// pentru a se putea identifica
private string FIREBOY_TAG = "Fireboy";
private string WATERGIRL_TAG = "Watergirl";

private string LAVAGEM_TAG = "Gem Lava";
private string WATERGEM_TAG = "Gem Water";

private MeshRenderer myMesh;
private SphereCollider myCollider;

[SerializeField]
private AudioSource GemAudio=new AudioSource();

bool onlyOnce;

Unity Message | 0 references
private void Awake() // se preiau componentele
{
    myMesh = GetComponent<MeshRenderer>();
    myCollider = GetComponent<SphereCollider>();
}

Unity Message | 0 references
private void OnTriggerEnter(Collider collision) // se foloseste fct „trigger” pentru
{                                               // a nu interactiona fizic
    if (collision.gameObject.CompareTag(FIREBOY_TAG) && this.tag == LAVAGEM_TAG)
    {
        Debug.Log("Fireboy has collected gem"); // mesaj
        GemCollector.LavaGemsCollected++; // se marcheaza drept colectat
        if(!GemAudio.isPlaying && !onlyOnce)
        {
            GemAudio.Play();
            onlyOnce = true; // pentru a nu putea colecta un diamant deja colectat
        }
        myMesh.enabled = false; // se ascunde
        myCollider.enabled = false;
    }
    else if (collision.gameObject.CompareTag(WATERGIRL_TAG) && this.tag == WATERGEM_TAG)
    {
        Debug.Log("Watergirl has collected gem");
        GemCollector.WaterGemsCollected++;
        if(!GemAudio.isPlaying && !onlyOnce)
        {
            GemAudio.Play();
            onlyOnce = true;
        }
        myMesh.enabled = false;
        myCollider.enabled = false;
    }
}

// spin animation

[SerializeField]
private float turnSpeed = 70f;

Unity Message | 0 references
private void FixedUpdate()
{
    transform.Rotate(Vector3.forward, turnSpeed * Time.deltaTime);
}
```

2. Apă, lavă și noroi



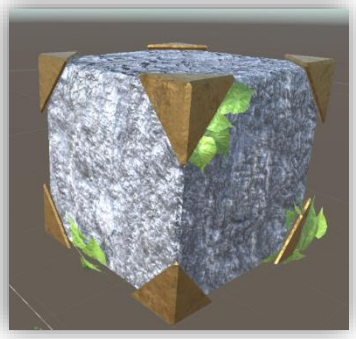
Animația corpurilor de apă, lavă sau noroi funcționează pe baza unei armaturi sub forma unei matrici de 5x5 „oase” care se contractă pentru a crea efectul de lichid.

Interacția cu jucătorul funcționează asemănător celei a diamantelor cu adiția cazului în care jucătorul moare.

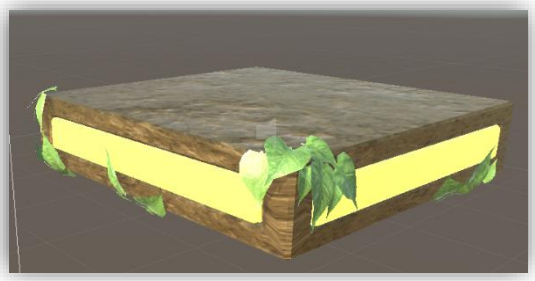
```
Unity Message | 0 references
13  private void OnTriggerEnter(Collider collision)
14  {
15      if ((collision.gameObject.CompareTag(FIREBOY_TAG) && this.tag == WATER_TAG)
16          || (collision.gameObject.CompareTag(WATERGIRL_TAG) && this.tag == LAVA_TAG) ||
17          (this.tag == GREENMUD_TAG && (collision.gameObject.CompareTag(FIREBOY_TAG) ||
18          collision.gameObject.CompareTag(WATERGIRL_TAG))))
19      {
20          // this is how to avoid static problem
21          FailLevel.sem = true; // se marcheaza ca nivel esuat
22
23          Debug.Log(collision.gameObject.tag + " has died."); // mesaj
24
25          // e important sa se stie care dintre jucatori a murit pentru a putea
26          // porni animatia corecta
27          if (collision.gameObject.CompareTag(FIREBOY_TAG))
28              FireboyMovement.Fireboy.isDead = true;
29          else
30              WatergirlMovement.Watergirl.isDead = true;
31      }
32  }
```

3. Bloc

Blocul funcționează ca un obiect solid care poate fi împins și pe care jucătorul poate sări.



4. Trapă



Trapa se poate mișca vertical sau orizontal, timp în care are efect sonor. Poate fi folosită drept o treaptă sau un blocaj spre altă secție a nivelului.

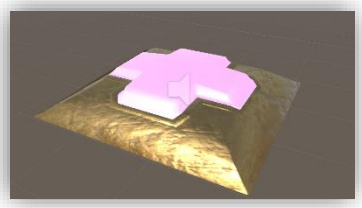
Pentru a activa deplasarea trapei este necesară activarea unui buton sau unei manete căreia îi este asociată (asociarea se face printr-o variabilă „indice” comună).

Culoarea pe care o emite trapa este aceeași cu cea a butolului/ manetei care o activează.

```
Unity Message | 0 references
33  private void FixedUpdate()
34  {
35      /*    Pentru ca un buton/maneta poate fi activata sau dezactivata inainte ca trapa
36       * sa isi termine deplasarea, se face o parcurgere care misca trapa in directia
37       * corecta in timp actual
38       *    Deplasarea se masoara in vectorul currentDistance si se compara cu distance
39       *    Directia este o variabila tip vector3 (contine valori pentru coord xyz) care
40       * decide daca trapa se misca orizontal sau vertical
41       *    Vectorul receiverIndex detine starea de activare a tuturor trapelor, din care
42       *    este verificat doar elementul indexului
43       */
44      if(currentDistance[index]<=distance && receiverIndex[index] != 0)
45      {
46          transform.Translate(direction * moveSpeed * Time.deltaTime);
47          currentDistance[index]++;
48          myAudio.Play();
49      }
50      if (currentDistance[index] > 0 && receiverIndex[index] == 0)
51      {
52          transform.Translate(direction * -moveSpeed * Time.deltaTime);
53          currentDistance[index]--;
54          myAudio.Play();
55      }
56  }
```

5. Buton

Butonul are două componente: baza și butonul propriu-zis, care se mișcă vertical la interacțiunea cu jucătorul. Are efect sonor pentru a semnala activarea acestuia.



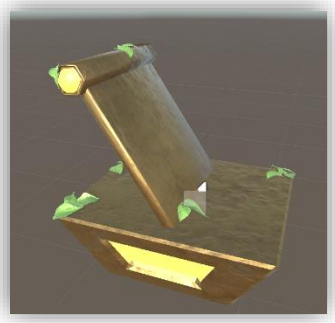
```
1 reference
void Pressed() // marcarea activari unei trape
{
    Debug.Log("Button has been pressed");
    prevPressedState = isPressed;
    TrapMovement.receiverIndex[index]++;
    onPressed.Invoke();
}

1 reference
void Released() // marcarea dezactivatii unei trape
{
    Debug.Log("Button has been released");
    prevPressedState = isPressed;
    TrapMovement.receiverIndex[index]--;
    onReleased.Invoke();
}
```

Conectarea butonului la o trapă.

Partea de dinamică pentru mișcarea butonului propriu-zis.

```
Unity Message | 0 references
void Start()
{
    Physics.IgnoreCollision(GetComponent<Collider>(), buttonTop.GetComponent<Collider>());
    if (transform.eulerAngles != Vector3.zero)
    {
        Vector3 savedAngle = transform.eulerAngles;
        transform.eulerAngles = Vector3.zero;
        upperLowerDiff = buttonUpperLimit.position.y - buttonLowerLimit.position.y;
        transform.eulerAngles = savedAngle;
    }
    else
        upperLowerDiff = buttonUpperLimit.position.y - buttonLowerLimit.position.y;
}

Unity Message | 0 references
void Update()
{
    buttonTop.transform.localPosition = new Vector3(0f, buttonTop.transform.localPosition.y, 0f);
    buttonTop.transform.localEulerAngles = new Vector3(0f, 0f, 0f);

    if (buttonTop.localPosition.y >= 0)
        buttonTop.transform.position = new Vector3(buttonUpperLimit.position.x,
                    buttonUpperLimit.position.y, buttonUpperLimit.position.z);
    else
        buttonTop.GetComponent<Rigidbody>().AddForce(buttonTop.transform.up * force * Time.fixedDeltaTime);

    if (buttonTop.localPosition.y <= buttonLowerLimit.localPosition.y)
        buttonTop.transform.position = new Vector3(buttonLowerLimit.position.x,
                    buttonLowerLimit.position.y, buttonLowerLimit.position.z);

    if (Vector3.Distance(buttonTop.position, buttonLowerLimit.position) < upperLowerDiff * treshHold)
        isPressed = true;
    else
        isPressed = false;

    if (isPressed && prevPressedState != isPressed)
        Pressed();
    if (!isPressed && prevPressedState != isPressed)
        Released();
}
```

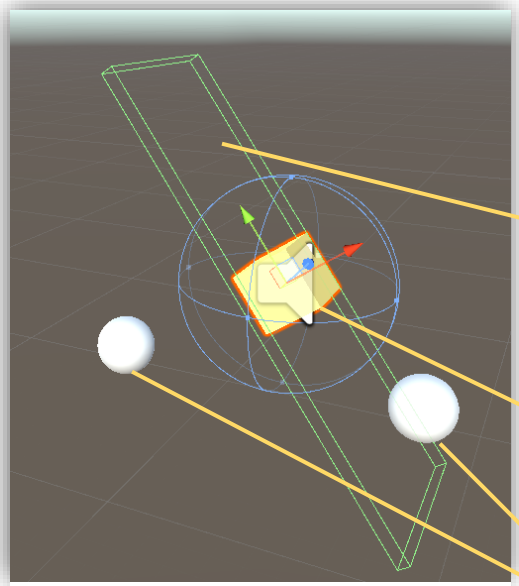
6. Manetă



Maneta funcționează asemănător butonului cu proprietatea adăugată de a putea activa o trapă și dezactiva alta în același timp.

Aceasta are două componente: baza și maneta care trebuie împinsă de jucător din oricare parte.

Are efect sonor la activare.



Mecanica din spatele unei manete

Aria de coliziune a obiectului care se rotește.

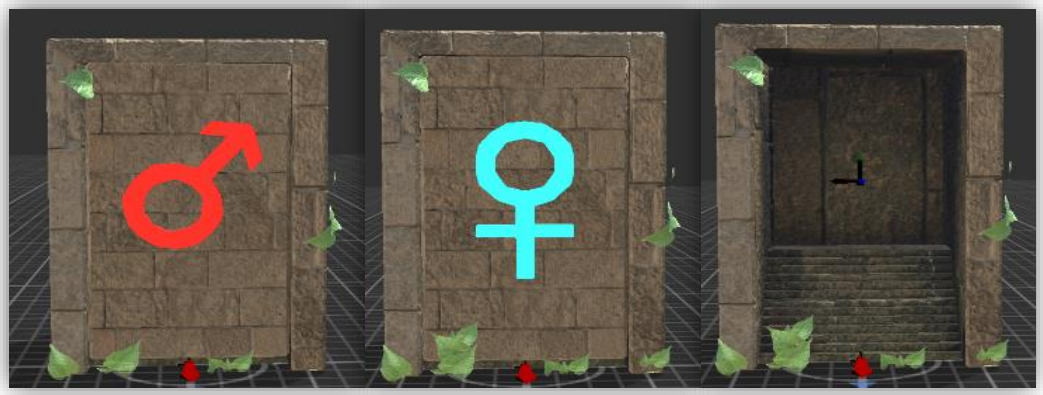
Obiectul care se rotește.

Sferele care, la coliziunea cu dreptunghiul, îi determină starea manetei de *on* sau *off*.

```
22  private void OnCollisionEnter(Collision collision)
23  {
24      if (collision.gameObject.CompareTag(ON_TAG))
25      {
26          Debug.Log("TrapDoor " + OnIndex + " has been activated and TrapDoor " + OffIndex + " has been deactivated.");
27          TrapMovement.receiverIndex[OnIndex]++;
28          TrapMovement.receiverIndex[OffIndex]--;
29          myAudio.Play();
30      }
31
32
33      if (collision.gameObject.CompareTag(OFF_TAG))
34      {
35          Debug.Log("TrapDoor " + OffIndex + " has been activated and TrapDoor " + OnIndex + " has been deactivated.");
36          TrapMovement.receiverIndex[OffIndex]++;
37          TrapMovement.receiverIndex[OnIndex]--;
38          myAudio.Play();
39      }
40  }
```

Activarea unei trape și dezactivarea alteia.

7. Ușă



Ușă Fireboy *Ușă Watergirl* *Ușă deschisă*

Ușa este mecanica finală a jocului. Aceasta se deschide când jucătorul pășește în fața ei (se închide la loc dacă acesta pleacă), iar nivelul se termină doar când amândoi jucătorii sunt în fața ușilor lor în același timp. Are efect sonor.

```
Unity Message | 0 references
void Update()
{
    if(finish) // daca nivelul a fost finalizat, se reseteaza animatia
    {
        anim.SetBool(DOOR_ANIMATION, false);
        return;
    }

    if (this.tag == FIREDOOR_ANIMATION && FireboyMovement.Fireboy.PlayerDestination == true)
    {
        anim.SetBool(DOOR_ANIMATION, true);
        if (!onlyOnce)
        {
            myAudio.Play();
            onlyOnce = true;
        }

    }
    else
        if(this.tag == FIREDOOR_ANIMATION && FireboyMovement.Fireboy.PlayerDestination == false)
        {
            anim.SetBool(DOOR_ANIMATION, false); // daca jucatorul pleaca din fata usii
            onlyOnce = false;
            myAudio.Pause();
        }


    if (this.tag == WATERDOOR_ANIMATION && WatergirlMovement.Watergirl.PlayerDestination == true)
    {
        anim.SetBool(DOOR_ANIMATION, true);
        if (!onlyOnce)
        {
            myAudio.Play();
            onlyOnce = true;
        }
    }
    else
        if (this.tag == WATERDOOR_ANIMATION && WatergirlMovement.Watergirl.PlayerDestination == false)
        {
            anim.SetBool(DOOR_ANIMATION, false);
            onlyOnce = false;
            myAudio.Pause();
        }
}
```

Pornirea/oprirea animației de deschidere

8. Jucători

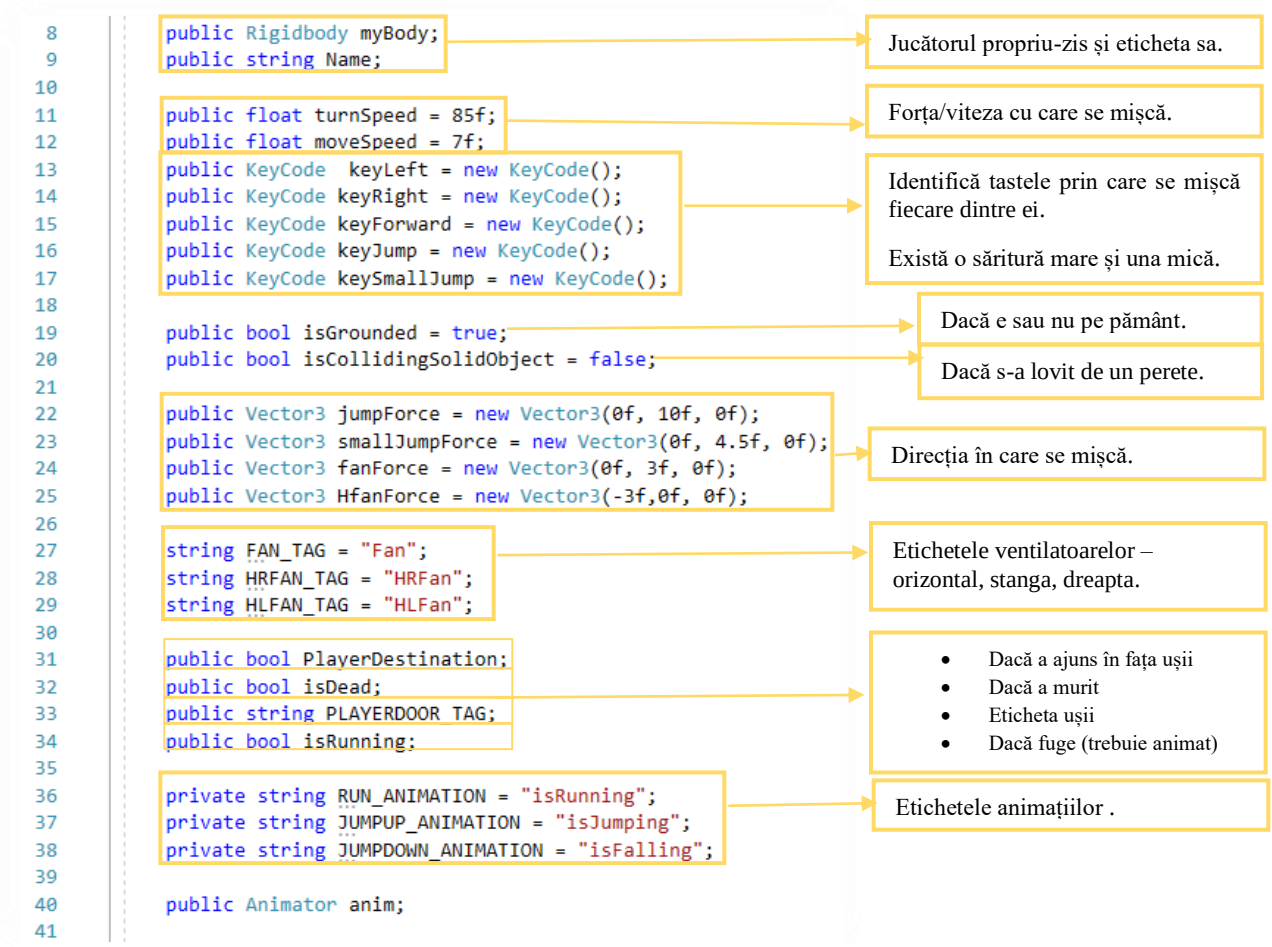
Jucătorii sunt:

- Fireboy (în dreapta ecranului) controlat prin săgeți (stânga, față, dreapta) și sare prin RightShift (mare) și RightControl (mic);
- Watergirl (în stânga ecranului) controlat prin AWD (stânga, față, dreapta) și sare prin Q (mare) și 2 (mic).

Aceștia interacționează cu restul obiectelor/mecanicilor și au efecte sonore pentru fugit, fugit prin apă/lavă, sărit și murit.

Fiecare jucător are o animație particulară pentru fugit, sărit, căzut și urcarea treptelor ușilor.

La nivel de cod, cei doi aparțin clasei „Player”, care realizează mișcarea, sărirea, interacțiunea cu un ventilatorul și cea cu ușa.



Variabilele clasei Player

Variabilele tip „public” sau [SerializeField] au valoarea ajustabilă în timpul rulajului.

```
42  public void Move()
43  {
44      bool sem = false;
45
46      if (Input.GetKey(keyForward))
47      {
48          if(!isCollidingSolidObject)
49              myBody.transform.Translate(Vector3.forward * moveSpeed * Time.deltaTime);
50          else
51              myBody.transform.Translate(Vector3.back * moveSpeed * 8 * Time.deltaTime);
52          anim.SetBool(RUN_ANIMATION, true); sem = true;isRunning = true;
53      }
54
55      if (Input.GetKey(keyRight))
56      {
57          myBody.transform.Rotate(Vector3.up, turnSpeed * Time.deltaTime);
58          anim.SetBool(RUN_ANIMATION, true); sem = true;isRunning = true;
59      }
60
61      if (Input.GetKey(keyLeft))
62      {
63          myBody.transform.Rotate(Vector3.up, -turnSpeed * Time.deltaTime);
64          anim.SetBool(RUN_ANIMATION, true); sem = true;isRunning = true;
65      }
66
67      if(!sem)
68      {
69          anim.SetBool(RUN_ANIMATION, false);
70          isRunning = false;
71      }
72
73      if (myBody.velocity.y > 0.5f)
74      {
75          anim.SetBool(JUMPUP_ANIMATION, true);
76          isRunning = false;
77      }
78      else
79          anim.SetBool(JUMPUP_ANIMATION, false);
80
81      if (myBody.velocity.y < -0.5f)
82      {
83          anim.SetBool(JUMPDOWN_ANIMATION, true);
84          isRunning = false;
85      }
86      else
87          anim.SetBool(JUMPDOWN_ANIMATION, false);
88
89  }
```

Mișcarea în față este făcută prin *translate*, iar drept consecință nu este oprită la întâlnirea unui obiect solid (perete sau ușă).

⇨ Semaforul isCollidingSolidObject îi indică ciocnirea cu un perete.

+animarea mișcării

Întoarcerea spre stânga sau dreapta.

+ animarea acesteia

Dacă s-a oprit din mișcare trebuie oprită și animația de mișcare.

Dacă velocitatea verticală este mai mare sau mai mică decât o treaptă de (+/- 0.5) atunci jucătorul este în cădere sau sare, deci trebuie animat.

Această animație nu ține neapărat de comanda de sărit, iar din acest motiv se află și în acest subprogram.

Subprogramul Move

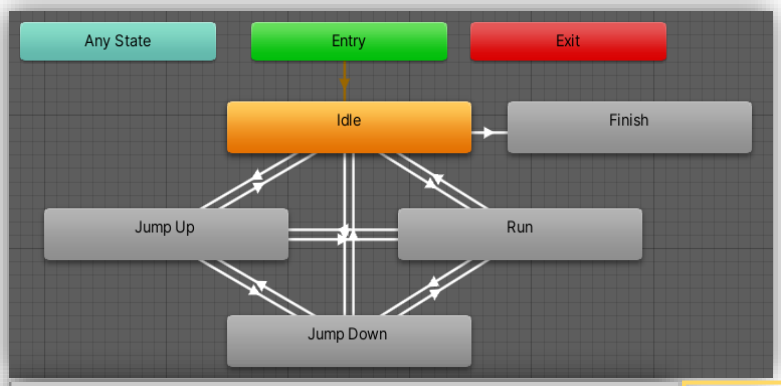
Subprogramele *Jump*, *Fan*, *ReachedDoor* și *LeftDoor* funcționează pe același principiu.

Clasa *Player* funcționează drept baza pentru cei doi jucători, apelată de scripturile FireboyMovement respectiv WatergirlMovement. Fiecare creează câte o variabilă de tip *Player* pentru care inițializează componentele necesare la momentul rulării programului.

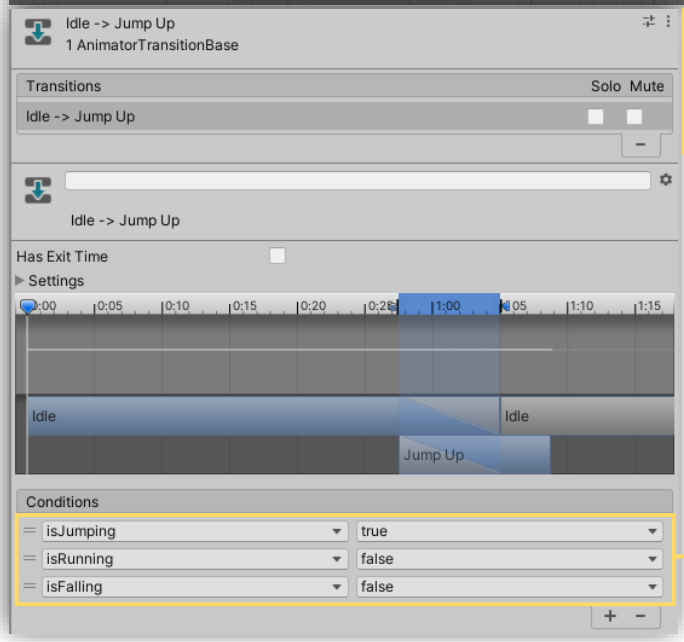
```
23  private void Awake()
24  {
25      //watergirl
26      Watergirl.Name = "Watergirl";
27      Watergirl.myBody = GetComponent<Rigidbody>();
28      Watergirl.anim = GetComponent<Animator>();
29
30      Watergirl.keyLeft = KeyCode.A;
31      Watergirl.keyRight = KeyCode.D;
32      Watergirl.keyForward = KeyCode.W;
33
34      Watergirl.keyJump = KeyCode.Q;
35      Watergirl.keySmallJump = KeyCode.Alpha2;
36
37      Watergirl.PLAYERDOOR_TAG = "Watergirl Door";
38
39      Watergirl.PlayerDestination = false;
40  }
```

WatergirlMovement și FireboyMovement verifică dacă jucătorul e în mișcare, dacă a murit și aspecte care țin de efectele audio.

Animarea jucătorului este facută de către o componentă *Animator* care folosește un *AnimationController:*



Imaginea de alături reprezintă AnimationController-ul. Acesta conține animațiile și determină tranziția de la o animație la alta pe baza unor condiții și semafoare.

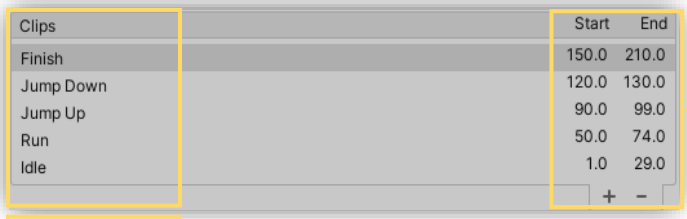


Imaginea de alături reprezintă tranziția și condițile trecerii din starea de repaus la cea de sărit.

Condițiile care trebuie îndeplinite pentru a se realiza tranziția.

Aceste variabile boolean sunt modificate în cod.

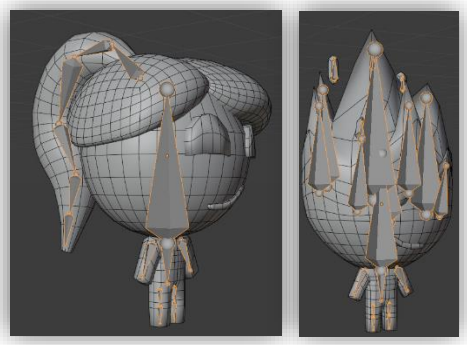
Modelele pentru jucători au fost realizate în Blender 3D. Fiecare jucător are o armatură/ un schelet potrivit modelului său. Animațiile sunt făcute pe un timeline din care, după ce modelele au fost importate drept .fbx, au fost decupate.



Secționarea propriu-zisă a animațiilor.

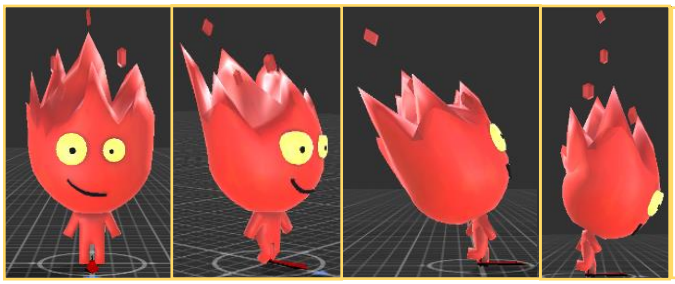
frame-ul de început => frame-ul de final

Numele animațiilor.



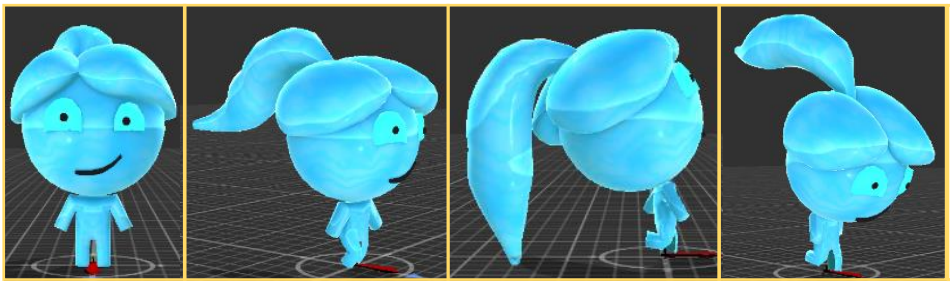
Modelele și wireframe-ul (geometria) 3d pentru jucători.

Armatura este delimitată cu portocaliu.



Animațiile pentru Fireboy

- ⇨ Repaus
- ⇨ Fugit
- ⇨ Sărit
- ⇨ Căzut



Animațiile pentru Watergirl

- ⇨ Repaus
- ⇨ Fugit
- ⇨ Sărit
- ⇨ Căzut

Script-ul *PlayerOnGroundCheck* este atașat obiectelor pe care jucătorul poate păși și de pe care poate sări. Are scopul de a asigura că acesta sare doar când este pe pământ (adică jucătorul nu poate sări dacă este deja în aer).

```
17      if (collision.gameObject.CompareTag(FIREBOY_TAG))
18      {
19          FireboyMovement.Fireboy.isGrounded = true;
20          FireboyMovement.Fireboy.anim.SetBool(JUMPDOWN_ANIMATION, false);
21          FireboyMovement.Fireboy.anim.SetBool(JUMPUP_ANIMATION, false);
22      }
23
24
25      if (collision.gameObject.CompareTag(WATERGIRL_TAG))
26      {
27          WatergirlMovement.Watergirl.isGrounded = true;
28          WatergirlMovement.Watergirl.anim.SetBool(JUMPDOWN_ANIMATION, false);
29          WatergirlMovement.Watergirl.anim.SetBool(JUMPUP_ANIMATION, false);
30      }
```

Analog, script-ul *PlayerCollideSolid* activează un Boolean care semnifică atingerea unui obiect solid (rezolvând problema creată de mișcarea *transform*). La activarea acestui Boolean, jucătorul primește un impuls care îl trimite cu o distanță mică în spate.

C. Cod

1. Timer

Script-ul *Timer* simulează o componentă care măsoară timpul (în minute și secunde) atât timp cât jocul nu este în stadiul de „pauză”. Această componentă este folosită pentru cronometrarea nivelelor.

```
Unity Message | 0 references
15  void Start()
16  {
17      startTime = Time.time;
18      isPlaying = true;
19  }
20
21
    Unity Message | 0 references
22  void Update()
23  {
24      t = Time.time - startTime;
25
26      if (isPlaying)
27          x+=t-f;
28
29      minutes = ((int)x / 60).ToString();
30      seconds = ((int)x % 60).ToString();
31
32      timerText.text = minutes + ":" + seconds;
33
34      f = t;
35
36      TimeFinish.text = minutes + ":" + seconds;
37  }
38
    2 references
39  public static void isPaused ( bool GiveValue )
40  {
41      isPlaying = GiveValue;
42  }
```

Preia valoarea nouă a timpului dacă jocul nu e în pauză.

Convertește timpul în valori întregi:

- ⇨ restul împărțirii la 60 reprezintă secundele;
- ⇨ partea întreagă a împărțirii la 60 reprezintă minutele.

2. StartMenu

Script-ul *StartMenu* conține subprogramele chemate de butoanele care aparțin părții de interfață și se ocupă de comutarea dintre „scene” (meniuri).

Pentru nivelul ternimat trebuie resetate variabilele.

```
9   public void PlayMainMenu ()
10  {
11      SceneManager.LoadScene("MainMenu");
12  }
13
    0 references
14  public void PlayLevel1 ()
15  {
16      SceneManager.LoadScene("SampleScene");
17      WatergirlMovement.Watergirl.isDead = false;
18      FireboyMovement.Fireboy.isDead = false;
19      FailLevel.sem = false;
20      DoorAnimation.finish = false;
21      GemCollector.LavaGemsTotal = 4;
22      GemCollector.WaterGemsTotal = 4;
23      GemCollector.LavaGemsCollected = 0;
24      GemCollector.WaterGemsCollected = 0;
25  }
26
    0 references
27  public void PlayLevel2 ()
28  {
29      SceneManager.LoadScene("Level2");
30  }
31
    0 references
32  public void PlayStartMenu ()
33  {
34      SceneManager.LoadScene("StartMenu");
35  }
36
    0 references
37  public void QuitGame ()
38  {
39      Debug.Log("Game has been exited.");
40      Application.Quit();
41  }
```

3. Nivel eșuat sau terminat cu success

Există patru script-uri care au scop la terminarea unui nivel: *FailLevel, DoorReached, GemCollector și ButtonColour*.

FailLevel

- ⇨ începe o „corutină” de 1.5 secunde pentru a permite începerea animației de fum;
- ⇨ Are efect sonor de „eșec”;
- ⇨ Activează meniul care permite restartarea nivelului sau întoarcerea la meniul de nivele.

DoorReached

- ⇨ semnalează faptul că nivelul a fost terminat cu success;
- ⇨ pornește două corutini pentru a arăta animația în care jucătorii urcă scările, iar apoi ușa care se închide în spatele lor;
- ⇨ pornește o a treia corutină care afișează meniul de final.

```
21    private void Update()
22    {
23        if (FireboyMovement.Fireboy.PlayerDestination &&
24        WatergirlMovement.Watergirl.PlayerDestination && onlyOnce)
25        {
26            StartCoroutine("FinishAnimation");
27            StartCoroutine("CloseDoor");
28            StartCoroutine("FinishLevel");
29            onlyOnce = false;
30        }
31    }
32
      0 references
33    IEnumerator FinishLevel()
34    {
35        yield return new WaitForSeconds(7.5f);
36
37        GemCollector.LavaGemsTotal = LavaGems;
38        GemCollector.WaterGemsTotal = WaterGems;
39        GemCollector.DecideGrade();//include time
40
41        Debug.Log("Level completed.");
42        Debug.Log("Finishing grade: " + GemCollector.Grade);
43        startAudio.Pause();
44
45        // show finish menu
46        FinishMenu.SetActive(true);
47        MenuButton.SetActive(false);
48
49        Timer.isPaused(false);
50    }
```

GemCollector

- ⇨ Are subprogramul *DecideGrade* care află nota pe care au obținut-o jucătorii, iar apoi o compară cu notele obținute înainte și o alege pe cea mai mare. Notele pot fi:
  - *O* dacă nivelul nu a fost jucat/terminat;
  - *C* (cea mai mică) dacă niciunul dintre jucători nu și-a colectat toate diamantele;
  - *B* (cea medie) dacă doar unul dintre jucători și-a colectat toate diamantele;
  - *A* (cea mai mare) dacă toate diamantele au fost colectate.
- ⇨ Afișează litera corespunzătoare notei primite în meniul de final prin funcția *Update* (care a fost folosită pentru că variabilele pe care le modifică sunt de tip *static)*.

ButtonColour

⇨ Colorează icoana de diamant a nivelului (din meniul de nivele) în culoarea notei obținute.

VI. Bibliografie și webografie

Texturi

- https://polyhaven.com/
- https://www.textures.com/library
- https://www.cgbookcase.com/

Unity

- https://www.youtube.com/watch?v=gB1F9G0JXOo&t=25791s – învățare unity
- https://www.youtube.com/watch?v=ogz-3r0EHKM&t=737s – blender-unity workflow
- https://www.youtube.com/watch?v=NjflKgMepQs&t=266s – importare din blender în unity
- https://www.youtube.com/watch?v=fTtLY0JdVqk – pentru crearea butonului
- https://www.youtube.com/watch?v=bh9ArKrPY8w&t=129s – mesh collider
- https://www.youtube.com/watch?v=FF6kezDQZ7s&t=4s – tranziți de animație
- https://www.youtube.com/watch?v=6OT43pvUyfY&t=109s – inserare audio
- https://www.youtube.com/watch?v=e94KggaEAr4&t=447s – tipuri diferite de mișcare a personajelor
- https://www.youtube.com/watch?v=b1uoLBp2I1w&t=723s – mișcare personaje
- https://www.youtube.com/watch?v=Au8oX5pu5u4&t=321s - mișcare
- https://www.youtube.com/watch?v=_QajrabyTJc&t=1s – mișcare la prima personană, atașare cameră de jucător
- https://www.youtube.com/watch?v=MFQhpwc6cKE&t=362s - atașare cameră de jucător
- https://www.youtube.com/watch?v=4HpC--2iowE&t=842s - atașare cameră de jucător
- https://www.youtube.com/watch?v=zc8ac_qUXQY – creare meniuri
- https://www.youtube.com/watch?v=28JTTXqMvOU&t=213s – creare hartă

Blender

- https://www.youtube.com/watch?v=At9qW8ivJ4Q&t=30s – tutorial pentru cunoștințe de bază
- https://www.youtube.com/watch?v=_C2ClFO3FAY&t=395s – animație
- https://www.youtube.com/watch?v=imbIsNAvUpM&t=30s – animație

Fireboy & Watergirl

- https://www.youtube.com/watch?v=h24akA8K-40 – poze de referință + inspirație
- https://www.youtube.com/watch?v=9IBQlR9opaI – muzică
- https://www.youtube.com/watch?v=qLH5ByH5TEg – efecte sonore
- https://official-fireboy-watergirl.fandom.com/wiki/Fireboy_%26_Watergirl_in_The_Forest_Temple – informații despre joc